1999-09

Distribution of firing directions in a coordinated surface-to-surface missile engagement

Utaaker, Inge Andre

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DISTRIBUTION OF FIRING DIRECTIONS IN A COORDINATED SURFACE-TO-SURFACE MISSILE ENGAGEMENT

by

Inge Andre Utaaker

September 1999

Thesis Advisor: Arnold H. Buss
Second Reader: James N. Eagle

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Master's Thesis

4. TITLE AND SUBTITLE
DISTRIBUTION OF FIRING DIRECTIONS IN A COORDINATED SURFACE-TO-SURFACE MISSILE ENGAGEMENT

5. FUNDING NUMBERS

6. AUTHOR(S)
Utaaker, Inge A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

The Norwegian Air Force and the Norwegian Navy both use the Norwegian developed Penguin surface-to-surface missile (SSM) but they use different tactics for launching it. The Navy recommends many attack directions, whereas the Air Force has the missiles approach the target area along a single axis.

This thesis investigates the effectiveness of different attack geometries using a discrete event simulation model that captures objects in motion, the detection of targets, the distribution of information, and the engagement procedures. The model includes ships, sensors, a data-link, missiles, missile batteries, air-target trackers, guns and the anti-air-warfare organization. Based on data from open sources, the simulation model of this thesis demonstrates that having all missiles approach the target area along the same bearing is the preferred SSM launch tactic under a variety of circumstances.

14. SUBJECT TERMS
Surface ships, Surface-to-Air Missiles, Surface-to-Surface Missiles, Guns, Radar, Air-target Trackers, Air-Defense Organization

15. NUMBER OF PAGES
142

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DISTRIBUTION OF FIRING DIRECTIONS IN A COORDINATED SURFACE-TO-SURFACE MISSILE ENGAGEMENT

Inge A. Utaaker
Commander, Royal Norwegian Navy
B.S., Norwegian Naval Academy, 1986

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1999

ABSTRACT

The Norwegian Air Force and the Norwegian Navy both use the Norwegian developed Penguin surface-to-surface missile (SSM) but they use different tactics for launching it. The Navy recommends many attack directions, whereas the Air Force has the missiles approach the target area along a single axis.

This thesis investigates the effectiveness of different attack geometries using a discrete event simulation model that captures objects in motion, the detection of targets, the distribution of information, and the engagement procedures. The model includes ships, sensors, a data-link, missiles, missile batteries, air-target trackers, guns and the anti-air-warfare organization. Based on data from open sources, the simulation model of this thesis demonstrates that having all missiles approach the target area along the same bearing is the preferred SSM launch tactic under a variety of circumstances.

TABLE OF CONTENTS

I. MISSILES AND MISSILE DEFENSE 1

A. HISTORY 1
B. WEAPON SYSTEMS 5
1. The Gun 5
2. Development from Guns to Surface-to-Surface Missiles 6
3. Surface-to-Air Missile Systems 8
4. Other Factors in Anti-Air-Warfare 11
C. TODAY 'S SITUATION 13
D. THE PROBLEM 14

II. PRINCIPLES FOR MODELING 15

A. THE QUESTION ASKED 15
B. MODEL DEVELOPMENT 17
1. Time Step and Discrete Event Simulation Models 18
2. Stages in Model Development 19
C. MODEL BALANCE 21

III. FROM TACTICS TO MODEL 23

A. OVERALL DESIGN AND BASIC PRINCIPLES FOR INTERACTION 23
1. Class Inheritance 25
2. Organization of the Defending Side 26
3. The Attacking Side 29
4. Neutral Instances 30
B. INFRASTRUCTURE 32
1. Simkit 32
2. Modkit 34

IV. THE KITCHEN PACKAGE 39

A. DEFINITIONS AND GENERAL ASSUMPTIONS 39
1. Units of Measure 39
2. Positioning 40
3. Success Criteria 41
B. MOVING OBJECTS 42
1. Two Dimensional Motion 43
2. Three Dimensional Motion 45
C. SENSORS AND TARGET DETECTION 46
1. The Constant Rate Detection Process 46
2. The Nonhomogenous Detection Process 48
3. The Sensor – Mover Geometry 50
D. PICTURE COMPILATION AND INFORMATION DISTRIBUTION 55
1. The `Contact` and `ContactManager` Classes 56
2. Target Prioritizing 57
3. The Data Link Procedure 58
4. Removing Targets 60
E. THE ENGAGEMENT PROCEDURES 61
1. Attacking with Surface-to-Surface Missiles 61
2. Selection of Firing Unit 63
3. The Trackers 66

4. Launching of Surface to Air Missiles *67*
5. Engaging with Guns *73*

V. EXPERIMENTAL DESIGN AND CONDUCT **77**

1. Split Plot Design *77*
2. The Primary Experiment *77*

VI. RESULTS **89**

A. SUMMARY STATISTICS AND GRAPHICAL DISPLAYS 90
B. ANALYSIS OF VARIANCE 97
1. The Main Experiment *98*
2. Analysis on Subsets of the Data *99*
C. SECONDARY EXPERIMENTS 101
1. Consumption of Surface-to-Air Missiles *101*
2. SSM Effectiveness Versus Salvo Size *104*
3. Attacking with Higher Time Concentration *105*
4. Attacking at Very Short Distance *107*
5. Air-Defense with Long Range SAM *109*

VII. CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK **113**

APPENDIX. THE CLASSES IN THE KITCHEN PACKAGE **117**

LIST OF REFERENCES **123**

INITIAL DISTRIBUTION LIST **125**

LIST OF FIGURES

3.1	Inheritance for Classes Modeling Motion	25
3.2	Inheritance for Classes Modeling Detection	26
3.3	Organizational Chart Defending Side	27
3.4	Organizational Chart Attacking Side	29
4.1	The Mover-Sensor Geometry	53
4.2	Queuing in the CIC	66
4.3	Listener Pattern in a SAM Engagement	70
5.1	Position and Formation at Time of Missile Impact	78
5.2	Launch Geometry with Intermediate Range and Medium Spread	79
5.3	Launch Geometry with Maximum Spread and Long Range	79
5.4	Launch Geometry with Minimum Spread and Long Range	80
6.1	Mean Number of SSM Hits for the Whole Plots	91
6.2	Mean SSM Hit over Spread, Performance and Distance	91
6.3	Mean, Pointwise and Simultaneous Confidence Intervals for Ranked Sub-Plots	95
6.4	Boxplot of SSM Hits over Spread	96
6.5	Boxplot for SSM hit over Distance, Policy and Performance	97
6.6	Boxplot of Fired SAMs over the Two Policies	102
6.7	Boxplot of Success Ratio for SAM over Firing Policy	103
6.8	Relationship between Salvo Size and Hit-Probability	105
6.9	Boxplot for SSM Hits over Inter Firing Delays 1 and 2.5 Seconds	106
6.10	Boxplot for Hits over Short and Super-Short Distances	108
6.11	Boxplot for SSM Hit over Long and Short Distance with Extended SAM Range	110

LIST OF TABLES

3.1	Class Organization	30
4.1	Edited Code for Creating Random Variable for Time to Detect	54
4.2	Methods for Providing the Distance Between the Sensor and the Mover	54
4.3	Method for Calculating a Units Availability	64
5.1	Parameters for the Missiles	82
5.2	Ships and their Combat Systems	83
5.3	Radar Data	83
5.4	Parameters for SAM Batteries	83
5.5	Parameters for Tracking Devices	84
5.6	Parameters for Gun Systems	85
5.7	Parameters for Decision and Classification Delays	85
5.8	Sample Detection Distances	86
6.1	Mean SSM Hit over all Sub-Plots	90
6.2	Coefficient of Variation for SSM Hit over all Sub-Plots	92
6.3	Sub-Plots Ranked on Mean SSM Hits	94
6.4	Function Call and Output from ANOVA with All Observations Present	98
6.5	Function Call and ANOVA Table for First Subset	99
6.6	Function Call and ANOVA Tables for Three Subsets of Data	100
6.7	Ratio of SAM Consumption per SSM Shot Down	103
6.8	Observations with Fixed Geometry, Varying Number of SSMs Launched	104
6.9	Function Call and ANOVA Table with Reduced Inter Firing Delay	107
6.10	Function Call and ANOVA for SSM Hits at Short and Super-Short Range	109
6.11	Function Call and ANOVA for Secondary Experiment with Extended Range SAM	110

EXECUTIVE SUMMARY

In the post World War II period the introduction of missiles represented the start of a fundamentally new and different era in surface warfare. Missiles are generally less plentiful than gunnery rounds, and higher emphasis must be placed on optimizing their usage. Still, tactics attempting to maximize the efficiency of surface-to-surface missiles (SSM) are not unique. In particular, tactics for the Penguin missile developed by the Air Force and the Navy in Norway use two different procedures for distributing the firing axis.

This thesis investigates the effectiveness of various attack geometries using discrete event simulation. A balanced model has been designed, taking advantage of two software packages developed at the Naval Postgraduate School: Simkit and Modkit. The focus is on three areas: (a) objects in motion; (b) detection of targets and distribution of target information; and (c) the engagement procedures. The model has as a core philosophy that disparate entities in nature appear as distinct entities in the code. The model includes ships, sensors (primarily radar), a data-link, missiles (surface-to-surface, semi-active and passive surface-to-air), missile batteries, air-target trackers, guns and the anti-air-warfare organization.

The user must supply the parameters that are necessary to run the model, thus keeping the model itself unclassified. Unclassified open source data were used to calibrate the model.

The main experiment examines the effectiveness of different degrees of spread on the firing axis. Other variables in the experimental design include the launch distance, the firing policy of surface-to-air missiles (SAM) by the defender (shoot-shoot-look (SSL) vs. shoot-look (SL)) and the performance of the air defenses on board the defending ships. It was found that for any specified launch distance, it was always beneficial for the SSM launcher to have all missiles approach the target area along one bearing.

Five secondary experiments were carried out, resulting in the following observations.

- The consumption of SAMs is significantly higher under firing policy SSL than SL, but SSL did not give significantly better protection to the ships under all circumstances.

- With increasing SSM salvo-size, the probability of hit for the individual missiles goes up, and there is a synergy effect with large salvoes. For example, increasing the salvo size by 50% would generally result in more than a 50% increase in the number of hits.

- Reducing the time between the first and last missile in the SSM attack increases the level of saturation of the defender air-defense and is beneficial to the SSM shooter.

- Varying the firing distance of the SSMs proved to have no significant effect in the main experiment when all the distances were outside the maximum range of the SAM. When one of the launch distances was adjusted to be inside maximum SAM range however, distance was significant.

- A subset of the main experiment was repeated with an extended SAM range. In this setup, the geometry with the shorter launch distance was inside SAM range. Again, the variations in launch distance showed significance.

ACKNOWLEDGEMENT

The author would like to thank Professor Buss for his constructive feedback when the model was designed and when the text was written. His patience and his ability to give the correct and the appropriate answer to any naive or advanced question is highly appreciated and it has made creation of this thesis a very educating experience.

I would also like to thank my parents who taught me the importance of doing my schoolwork, my wife Gunn for her quiet but continuous and vital support and finally my children who are appreciating my view on the importance of doing their schoolwork.

I. MISSILES AND MISSILE DEFENSE

Naval Warfare consists of many different tasks that may be assigned to Naval Forces. Two subsets of Naval Warfare tasks are anti-surface-warfare and anti-air-warfare. These two warfare areas involve, respectively, the strategies, tactics, and procedures employed when one naval unit engages a target on the surface of the ocean or in the air. The nature of anti-surface-warfare (ASuW) may be offensive as well as defensive, while anti-air-warfare (AAW) is generally a defensive operation. The two types of operations are linked together: when one force engages another it is ASuW, whereas the defense carried out by the target units would typically be an AAW task. In both types of warfare a variety of weapons are employed, but the most important weapon is generally a missile.

Missiles represent a scarce resource, and it is crucial for any commander at sea to have the best possible guidelines on how to employ them. For this reason tactics have been developed to optimize use of the weapon systems. Since missiles are expensive and therefore not fired for practice very often, data from real use are sparse, so development of tactics largely relies on theoretical studies. This thesis is such a study. A discrete event simulation package has been designed to investigate the use of missiles in general and in particular to compare and contrast two different tactics for employment of surface-to-surface missiles.

The rest of Chapter 1 will discuss the history of the missile as a weapon and describe the basic principles for its operation. The chapter will also address systems and procedures related to the operation of launching a missile attack and to the air defense.

A. HISTORY

> They saw from the direction of Port Said a green rocket and right after it a streak approaching them. They had time to increase their speed, to turn the ship around and fire a few bursts of the machine gun at the missile. Its course, at first, was not headed directly at the ship. At a certain point, apparently when its homing device detected the ship, it changed course and smashed in. A minute or two later, another missile arrived and hit the engine room. (New York Times, October 23. 1967)

The article reports the Egyptian sinking of the Israeli destroyer "Elath", the first known successful use of surface-to-surface missiles. The Egyptians were supplied with high technology weapons from the Soviet Union and had the Russian "Styx" missiles in their arsenal, designed for launch from their patrol boats of the Komar class. Later in the same article Commodore Erel of the Israeli Navy is cited: "What lessons did we learn? We learned that we need the proper tools – and there are proper tools." We do not know today, almost 32 years later, specifically to what the Commodore was referring, but obviously better protection from incoming missiles must have been one of his thoughts.

The development of combat units at sea for the last hundred years, through the end of the Second World War, was mainly concerned with the size, the building materials and the propulsion of the ships; and the new concepts achieved in the period proved to be substantial. However, this evolution did not result in comparable improvements in ships' armaments. The main weapons continued to be the guns and, although ship artillery had developed into a refined stage, the basic physics still was the same. In the following we will discuss why the introduction of missiles represented such a milestone in the evolution of naval warfare.

For the naval warship, the introduction of an effective power plant and the propeller clearly represented a revolutionary improvement in maneuverability and speed. Likewise, the introduction of steel represented a leap in the development of a "man of war." The means of obtaining data of the whereabouts of the enemy, the sensors, also had a dramatic improvement with the introduction of the radar. The most significant changes in the inventories of naval weapons up to 1945 were the introduction of the submarine, the torpedo and the aircraft carriers. Although the first real submarines, often referred to as "true submersibles," were not seen until the nineteen fifties, the submarines used in World War II had high impact on the war despite their limited speed and endurance. The guns were, however, by far the most important anti-ship weapons on surface combatants. The guns had, of course, gone through significant improvements in loading mechanisms, caliber and range, accuracy and rates of fire.

In a broad sense the development of gunnery had two major aspects. First, there was the "guns versus armor" race, where a heavier gun would trigger thicker armor. But thicker armor would trigger even heavier guns. The main goal was to deliver heavier shells than the enemy could endure, but still be able to survive hits from the opposing forces. This race culminated with the introduction of the Iowa class of battleships commissioned in World War II by the United States Navy.

Second, and less "brute force" oriented, it was imperative to achieve the first few hits before the enemy could respond with a counter-action. This could be achieved by having superior range, more guns than the enemy, or higher rates of fire.

The desired reduction in the number of hits necessary to disable, and the goal of being superior in range, called for larger and heavier shells, which in turn implied larger and heavier cannons. Larger caliber guns however usually resulted in reduced rate of fire and fewer barrels available on each ship. There were conflicting demands and limited possibilities for further development of the traditional artillery because obtaining accurate target location data for the maximum range of the largest guns was a very difficult task. By the end of World War II, the state of the art in gunnery could only be accomplished in the very largest warships.

Naval warfare in World War II also shifted from the battlegroup versus battlegroup gunfire exchange. The most direct heir of the traditional battles was to be fought in the air with ship-carried aircraft on both sides. This kind of warfare, where the surface combatants mainly stayed out of gunnery range, was seen primarily in the Pacific Ocean. The second new direction involved the use of submarines and submarine defenses. Anti-submarine warfare, or at least protection of merchant shipping from submarine attacks, was the major challenge in the Battle of the Atlantic.

While ship-to-ship artillery was given low priority in the post war decades, gunnery kept its importance as the main ingredient in the self-defense of ships from airborne attacks. At the time of the sinking of the "Elath," more than a quarter of a century had elapsed since the last major improvement in the area of artillery. The only

new construction of a major seagoing gun-carrier was the Soviet "Sverdlov" class cruiser, which was based on a pre-World War II German design.

While the development of gunnery all but stagnated during and after World War II, the same period brought the era of the missile. The German V1 and V2 rockets were operational in the last half of the war, and in the following decades the concept of missiles rapidly found applications in ground-to-ground combat, ranging from the smallest battlefield anti-tank missiles to the major carriers of nuclear deterrence. Missiles and rockets also took on an important part in air-to-ground and in ground-to-air warfare. The 1970-71 edition of Jane's Weapon systems (probably the most important source of unclassified military data) lists numerous systems of the kinds mentioned above, and they appear to be developed to a high degree of sophistication. There are however only five listings of missile systems in the area of naval ship-to-ship weapons, among them being the Soviet "Styx" missile and the "Gabriel" of Israel, a system a few years younger. Several systems are described as being under development. There are listings of seagoing missile-systems for anti-air-warfare, some of whom have an anti-shipping capacity as a secondary role.

Throughout history, any introduction of a means of conducting war has led to the development of counter measures. This "law of responses" applies to every aspect of warfare including tactics, weapons and training and may result in either passive or active measures. Passive responses include body armor and low-signature aircraft, while active responses include the transmission of radio noise to disturb enemy communications and the surface-to-air-missile (SAM). At the beginning of the 1970's anti-missile-missiles were being designed, probably because it was recognized that the early anti-air-missiles would be ineffective against the new weapons, namely the SSMs. We have reasons to believe that the sinking of "Elath" speeded up this process considerably. This is indicated in Jane's under the description of two systems under development: "For the successful interception of an incoming anti-ship-missile great accuracy and extremely short reaction time will be required." (Jane's Weapon Systems 1970-71, Sea Wolf) and "assembled on

urgent basis from existing hardware." (Point Defense Missile System). The latter system was developed further and is today known as NATO Sea Sparrow.

B. WEAPON SYSTEMS

In this section a selection of the principal components of the weaponry of a modern warship will be discussed. The systems described will be those important for this thesis, anti-surface and anti-air warfare.

1. The Gun

The basic dynamics of bringing explosives to the enemy by using a gun have remained unchanged for centuries and can be illustrated as follows. First the firing unit obtains accurate information about the distance and direction from the gun to the target. Second, information about target motion should be taken into the calculations, as the necessary accuracy will increase proportionally to the firing range. The latter data will make it possible to fire the round toward a predicted future target position, compensating for target advance during time of flight for the projectile. Finally the gun has to be elevated, trained correctly and the round fired.

Since heavy artillery rounds easily exceed one minute of flight time, no matter how precise the calculations of gun training and elevation, a hit still depends on the target unit not changing its course or speed during the time of flight. This is due to the simple fact that from the moment the shell has left the barrel it is subject only to the forces of wind, weather and gravity. The firing unit can neither control the round, nor is any means of self-guidance available. This fact is well understood, and basic gunfight tactics recommend high speed and frequent changes of heading to complicate firing solutions for the attacker.

These factors make pure chance a major contributor to the outcome, since the probability of a hit with a single round is low and tends to fall with increased range and

maneuverability of the target. In the 1990's, ammunition was introduced with some guidance capabilities, but such equipment is still very rare.

2. Development from Guns to Surface-to-Surface Missiles

The introduction of the first surface-to-surface missiles (SSM) significantly changed the anti-surface-warfare in several ways. The deadweight versus payload ratio was dramatically altered, the probability of hit for a single round was increased substantially, and it became possible to continue the growth in maximum range for surface-to-surface weapons that appeared to have halted with the introduction of the 40.6 cm (16 inch) guns. Thus, a fundamental change in the conduct of anti-surface-warfare was brought about.

a) Weight

In order to deliver a shell of a given weight from a gun, it is necessary for the ship to carry more than 100 times that weight in the gun itself. For example, the barrel alone of a 3-inch gun weighs more than 800kg, and the weight of one shell is around 6kg. If the gun were constructed with armor to give protection from an equally heavy enemy gun, the weight ratio would be even less favorable. In contrast, the early SSMs carried a warhead with a mass of about one fifth the weight of the entire system. Obviously this improvement represented a new era; even from a 200-ton patrol boat it became possible to fire very lethal weapons. Prior to implementation of the SSM, a smaller craft such as a patrol boat had no means of threatening a larger unit apart from a torpedo. The torpedo was the weapon that made it feasible to keep patrol boats in the modern navies, and it had many of the same advantages as the missile when it came to payload versus deadweight ratio. However, its accuracy and range was generally poor until the introduction of homing devices and wire guiding. The SSM made it possible to maintain relatively powerful weapons onboard small units and dramatically increased the possible range and accuracy of weapons delivered by these platforms. During the first years of SSMs, their

advantages were first appreciated by smaller navies, by nations traditionally operating smaller units. These nations includes Norway, with the "Penguin" missile and "Storm" class Fast Patrol Boat, and Israel with the "Gabriel" SSM on board French built patrol boats.

b) Target Information

The introduction of SSMs significantly relaxed the need to acquire accurate target data. While exact knowledge of the target's position, course, and speed is mandatory when firing a gunnery round, the missile can either be controlled or can control itself during the time of flight, reducing the requirement for accurate target data. Almost all SSMs have their own built-in system of sensors. The highly sophisticated sensors used in modern missiles can broadly be divided into two categories: passive, homing on infrared or lower frequency electromagnetic radiation, and active, where the missile has its own radar or equivalent device. Independently from the type of sensor, the SSMs are typically equipped with a three-phase guidance system, briefly discussed next.

(1) Phase One. This phase starts before the launch when the missile is given a route to follow. Depending on the sophistication of the missile, the aiming point can be the position of the enemy at the time of the firing or it can be an approximate future position of the target. For the most advanced systems, the route can be a set of waypoints for the missile to navigate through before starting the final leg toward the target. After launch, the weapon will move according to this programmed route until the second phase of the guidance starts.

(2) Phase Two. The second phase of the missile flight starts when the sensor is activated. There are large variations in the programmed time of seeker activation and the geometry of the search to be conducted. But missiles with a radar seeker are generally able to have search patterns that are larger and further ahead of the missile than those with passive infrared seekers. Weapons designed for use in littoral waters will typically have smaller search areas than those designed for open water attacks

in order to reduce the risk of land masses appearing as false targets inside the search areas.

The missile will continue to search for its target as long as its seeker is active or until it detects signals that can be distinguished from the background noise and accepted as the true target.

(3) Phase Three. The third and final phase of the missile flight starts when the sensor detects an object in the search area and accepts it to be the target. At this point the SSM will abandon its preprogrammed route and will start guiding itself using continuously updated information about target position and motion. This is often referred to as the homing or lock-on phase. The missile will alter its course as needed to hit the target ship. For some systems this is taken a step further so that the seeker also determines a preferred point of impact in the target ship. The size and geometry of the search area and the ability to update target information while approaching compensates for inaccuracies in the target position held by the firing platform.

c) Range

As a rule of thumb, the efficient range of a gun is about one kilometer for every centimeter of caliber. For today's modern systems this is too modest, and a realistic factor might be between 1.5 and 2.0. The maximum range for the 12.7-cm (5-inch) gun in the modern United States warships is estimated to be 23 km (Jane's Fighting Ships 1998-99, Arleigh Burke class). Still, the range of the medium sized gun was already exceeded with the first generation of SSM, and today SSM effective ranges vary from 27 km (the Norwegian built Penguin MK2) to 130km (US-built Harpoon) (Jane's Fighting Ships 1998-99). Arguably, the larger maximum range of surface weapons was the most significant change brought about by the implementation of the SSM.

3. Surface-to-Air Missile Systems

As mentioned in Section A, there had been SAM systems operational for some time when the developments of the first SSMs were completed. However, these SAMs

were designed to counter the threat from aircraft. Although defense against SSMs and defense against aircraft are similar in many respects, the anti-missile defense was recognized early on as much more demanding. There are many reasons for the increased difficulty, resulting primarily from a radically reduced reaction time for the defenders.

Normally, detection of aircraft and SSMs are both done using the same equipment, the air warning radar system. There are two critical parameters determining the range at which it is possible to discover a flying object on radar: the object's radar cross-section (RCS) in the frequency of the radar, and the object's altitude. The physical size, the shape and the electrical conductivity of the surface of the object are the main determinants of radar cross section. Since an SSM is generally smaller and more streamlined than an aircraft, its RCS is normally smaller.

Furthermore, from the first SSMs, it has been a goal for the designers to make the weapon able to fly at very low altitudes, hence the expression "sea skimming." Sea-skimming affects the detection range in two ways. First, any target will be invisible to radar as long as it is below the radar horizon, and the range to the horizon decreases with decreasing altitude. Seond, a radar will always receive unwanted reflections from the sea, and the lower a missile flies the harder it will be to discriminate it from this noise.

The afore mentioned differences between aircraft and SSM result in a change in one major factor of the air-defense: drastically shortened reaction times for the anti-air-warfare organizations. The shortened reaction times necessitated new weapons, as well as revision of tactics and training. Finally it probably also required revision of the overall threat assessment for seagoing military forces.

Surface-to-air missiles (SAMs) span a large variety, ranging from hand-held, very short range anti aircraft weapons, to large, long-range systems capable of engaging practically any air threat. The discussion here will be limited to a brief description of the principles of systems capable of countering an incoming SSM.

The SAM will normally not be launched unless the firing platform holds accurate data on the target. Unlike the SSM, most SAMs cannot be given a "general" target-area and then find the actual target position on their own after launch. Consequently the first step in a normal engagement sequence is the determination of the target position. If the SAM-equipped ship is operating alone, this will be done by its own sensors. On the other hand, if it is operating in a group of friendly units, target information will be obtained either by own sensors or by some other unit and transferred by use of a data link. After detection of a threatening target by one unit in a force, the same information will be disseminated to all the individual command and information systems within a few seconds. If the target is detected outside a minimum range, the officers in command of the group decide which unit should deal with the threat or the decision could be made by an automated system. If the target is detected too close to allow time for such considerations, the unit threatened will act in self-defense and engage the target without further orders. When it is determined which unit is responsible for the threat, the next step will be for that ship to transfer the target data from the combat information systems to a tracking system.

Tracking of the target is the process of having continuous knowledge of the target position and velocity, and can be done according to one of three different approaches or by a combination of them. In an all-passive system, the tracking device will focus in the direction and elevation dictated by the command system. The missile will be ready for launch when its internal sensors detect the target. This approach is most common with short range SAMs. For intermediate range systems it is common to have semi-active missiles. In this approach, there is an active radar tracker working, fixed on the firing ship and often referred to as the illuminator. This tracker will search along a general bearing to the target specified by another system. When it finds and radiates toward the target the SAMs to be fired will detect some of the reflected energy and the system is ready for launch. During the flight of the SAM the ship must maintain the illuminator pointing at the target, and the missile will steer toward the radar signals until it hits the source of the

reflections, the target. For long range SAMs, a third method is used most often. Here the missile carries its own active sensor, which tracks the target for a large portion of the flight time. This missile guidance technique is similar to that used in the final phase of an active SSM approaching its target.

4. Other Factors in Anti-Air-Warfare

In anti-air-warfare (AAW), the various SAM systems are the key elements in the protection of ships from attacks through the air. However, like almost any well-designed military defense, AAW is constructed of various types of instruments organized in layers. Typically, if the force does not carry its own, organic, aircraft, SAMs represent the outer layer of the AAW defense. Groups of ships with aircraft carriers included will normally have patrolling fighter planes as their most distant AAW band. As we will discuss later, SAMs are a scarce resource, and hence their deployment is considered a force responsibility. The implications of a weapon being a force asset is primarily that the commanding officer of a single ship will not fire a SAM at a target without coordinating his effort with the authority controlling these resources. An important exception to this rule is an act of self-defense in which no demand for coordination prevails. Deployment of organic aircraft is also subject to force coordination.

The weapon systems to be discussed in this sub-section are not force assets, meaning they are not as rigidly controlled by the task group commander as the SAMs. They have a shorter radius of operation and are available in larger numbers than the surface-to-air missiles. The spectrum goes from "hard-kill" to "soft-kill" tools, that is from guns with anti-air capacity through close-in weapon systems to various physical decoys and electronic deception.

a) Hard Kill

The "hard-kill" weapons are all guns of various ranges and rates of fire. As discussed in Sub-section B.1 (The Gun), long range guns mean large caliber and low rate

of fire, while shorter range guns allow for increasing the rate of fire. Many modern frigate-sized and larger ships carry a 3 to 5 inch (76 to 127mm) gun that has anti-air capacity with effective range in this mode going out 6 – 8 kilometers. Their maximum rate of fire typically ranges from 30 to 120 rounds per minute. Their probability of hit and effective ranges are not fixed numbers, but rather variables depending on parameters such as the target motion, altitude, and size. The probability of hit, obviously, goes up as the target gets closer to the gun if everything else is equal. On the other hand, the angular speed of the target is also a contributing parameter. In fact, the probability of hit might very well be higher on a target at 5 km than one at 3 km if the first one is travelling directly at the shooter.

The close-in weapon systems (CIWS) are also guns, but with caliber in the range of 20mm to 30mm. These systems are constructed to have extremely high rate of fire, some as high as 100 rounds per second. The range is short, rarely exceeding 4km, and it is common to have the necessary sensors located directly on the gun mount.

b) Soft Kill

Soft kill weapons refer to equipment that is effective without having a physical interaction with the target. Decoys are devices deployed around the ships to create false targets. The intended result is to make the enemy fire weapons on the decoys rather than the ships. If the decoys are activated after launch of enemy SSMs, their purpose is to prevent the SSM from acquiring a target, make it acquire a false target, or even fool it to jump from a ship lock-on to a trap. Decoys designed to spoof radar are commonly a "ball" of lightweight radar reflecting particles launched by special mortars ("chaff"), while devices to lure infra-red sensors are composed of ignited metal burning intensely at high temperature, flying or floating at some distance from the ships.

Finally, "soft-kill" may be accomplished electronically. Electronic deception may be employed to accomplish the same result as decoys, namely to confuse enemy firing solutions or distract incoming missiles. This may be done by creating false targets electronically or by transmission of noise.

C. TODAY 'S SITUATION

We have argued that the introduction of the SSM created a fundamental change in the development of anti-surface-weapons. It appears, however, that to some extent the pendulum has swung too far away from the guns and traditional artillery. Some classes of warships have been designed totally without gun systems, but in retrospect these are generally judged to be sub-optimal solutions. The Royal Navy type 22 batch II frigates are examples of highly sophisticated vessels not equipped with caliber weapons. Although designed mainly for open water anti-submarine warfare, there are situations where a gun is more effective than a surface-to-surface missile. Two such situations might be shore bombardment, which cannot be done with SSMs, and low intensity conflict, where use of SSMs might result in unwanted warfare escalation.

Most frigate-size and larger combatants built in the 1990's have a gun for low intensity situations and as a secondary weapon in both surface-to-surface and surface-to-air combat. SAMs and SSMs however, maintain their positions as the main weapons for AAW and anti-surface-warfare (ASuW) respectively. Traditionally the numbers of missiles ready has been limited to 8 to 16 for SAMs and 4 to 8 for SSMs. Some of these units have extras available in storage, but reloading is a lengthy process, and the extra weapons will probably not be available in the same incident. There are however both Russian and US ships with considerably larger numbers of missiles ready for near immediate use, and other navies are following the same path. What is new is the ability to launch missiles from a vertical position, making it feasible to have a much larger number of weapons ready with very short reaction time. Furthermore, the possibility of using the same launching equipment and storage arrangements for missiles against targets in the air, on land, on the surface and under the surface has lead to greatly improved efficiency and flexibility.

That being said, however, missiles of any kind remain a scarce resource. Not only are missiles available in very limited numbers for the commanders at sea, they also represent state-of-the-art technology and are very costly to develop, purchase and

maintain. Firing a missile is a rare and expensive operation. For these reasons, substantial resources are put into continuously evaluating tactics and procedures on how to best exploit both the SAMs and the SSMs. Also the training and exercise programs for ships is heavily influenced by the desire of ensuring the best possible utilization of the costly weapons. For AAW especially, extensive drill is required to achieve the coordination necessary for operational success.

D. THE PROBLEM

This thesis investigates the process of defending a group of ships at sea from incoming surface-to-surface missiles. A discrete event simulation model is developed to compare different tactics to defend the ships under attack. The simulation produces a measure of effectiveness for the various tactics, using the parameters specified by the user.

The remainder of this thesis is organized as follows: Chapter II will discuss modeling in general. Chapter III will give the overall description of the simulation model created in this thesis, the `kitchen` software package, while Chapter IV will address in more detail some of the assumptions and algorithms. Chapter V will describe the principles and parameters, and Chapter VI will discuss the results of the experiments. Finally, Chapter VII will summarize the experimental the results of and give suggestions for further work in this field.

II. PRINCIPLES FOR MODELING

In the previous chapter we discussed the challenges that face decision-makers when they plan and execute a missile attack and what the officers in the defending ships must do to counter the missile threat. Clearly any missile is a highly valuable item in a limited inventory. When action is about to take place, however, the value of a missile is not related to cost as much as to its contribution to the mission. In short, it is imperative that the missiles at hand are used optimally; in fact it may very well be a question of life and death for the personnel involved.

Optimizing the use of any device requires understanding the physical surroundings as well as knowing the possibilities and restrictions built into the instrument. Military equipment in general, and the missile in particular, is no exception to this rule. In fact missiles are so rarely used that the theoretical understanding of how they work and may be utilized is more important than with most other kinds of equipment. To navigate a ship or to operate a computer, involved personnel must develop their skills both through theoretical studies as well as practice. The same holds only partially for missiles, because only a fraction of the skills acquired by involved operators and decision makers come from actual firings. Most of the qualifications are developed by theoretical studies, by various kinds of simulations, and finally practices short of actual firing of weapons.

The rest of this chapter will outline the question we are trying to answer and describe the overarching philosophies that were in the background when the simulation package was designed.

A. THE QUESTION ASKED

Operating procedures are developed for all almost all tasks that need to be accomplished by military personnel. For employment of military equipment, such guidelines are commonly referred to as tactics. As with education, development of tactics

will rely on practice when possible, but will have to depend on theoretical analysis when necessary. When particular tasks or operations are easily exercised and frequently practiced, we tend to be confident that the tactics are effective. With limited actual data, or only data from exercises, however, we do not know to what degree tactics are realistic. It is hard to establish strong evidence that everything important is captured.

If two different agencies each develop tactics designed to address the same problem and the tactics turn out to be partially contradicting, there may be reasons to claim that one of them is not correct. In this case there is reason to believe that only one of them gives the best description of how the current problem should be solved. The starting point for this thesis was the discovery of such a difference in the tactics for employment of the Penguin anti-shipping missile in the Norwegian armed forces. The available data in this area are from some very sparse exercises, and essentially no real situation data. Hence the existing tactics are based on theoretical studies and computations.

The Royal Norwegian Air Force and Navy both have the Penguin missile in their inventory. The missile is designed with an infrared passive homer and optimized for use against medium and small ships in confined waters. Without revealing classified details we will go no further than to establish that the two services recommend different courses of action to maximize the probability of a successful attack. The Navy recommends geographically spreading out the firing platforms and also the use of waypoints, to have the SSMs approach the target from many directions. The idea is to create an almost omnidirectional threat to the enemy, making it impossible for him to cover all angles of attack. On the other hand, when launching Penguins from F-16 fighter aircraft the Air Force recommends concentrating the firing platforms and having all missiles approach the target area from one direction.

We will not claim that the lack of similarities in the two tactics results from a totally different understanding of how the Penguin missile or an air-defense works; differences between ships and aircraft may partly explain the disagreement. Even if everything else were equal, the two services would still apply different tactics. This study

will examine the *pro et contra* of both tactics under the assumption that the officer in tactical command may choose her attack geometry freely - attack geometry being the various combinations of firing axes and launch distances. If it turns out that one of the tactics is superior, the agencies of both services should examine the assumptions under which the model is built and the results produced. If the assumptions are acceptable and their tactic is in conflict with the model's prediction of optimal use of SSMs they should probably make changes to their guidelines to the user of the weapons. If there appears to be no significant difference between the two tactics however we may conclude that both parties are using near optimal solutions.

The question asked in this thesis, then, is: Is it possible to say that one tactic is superior to the other; and if so, under which circumstances?

B. MODEL DEVELOPMENT

It is important to understand that every model has shortcomings when compared to "the real world." The art of modeling deals with two major challenges: Which relationships should be modeled and to what level of detail. The model should not necessarily try to capture as much of reality as possible. Only the important parts of nature should be captured, those parts important to the question being asked. In fact, the issue of constraining the level of detail is so important that making this decision correctly when designing the model may determine if there will be a model at all.

What is important to model then? The answer to this question hinges on the intended use of the model. Two models of one ship, for example, will have radically different features if the first model will be used for navigational training and the second is for basic training of the firemen in the ship's crew. Once we have decided that we are going to build a fireman's model of a ship, the next step would be to decide on the level of detail to capture. To assume that the model can be effective if it does not present the true positions of the firehoses would be an oversimplification, while insisting on the correct color of wallpainting is probably unnecessarily detailed.

With these seemingly obvious but nevertheless very important principles stated, we will now describe the two principal ways of designing simulation models.

1. Time Step and Discrete Event Simulation Models

There are two mechanisms for advancing the time in simulations: Time step and event step. Time step simulations advance time in fixed increments, and after every step the model will update each component's state variables. For example, a component simulating motion would have its position, speed and heading updated for every increment of simulated time. One of the problems addressed in this thesis is the interaction between a radar and a surface-to-surface missile. If this had been done with fixed time step simulation with increments of one second, say, it would require asking the SSM every second about its position and altitude, and then pass this information to the entity simulating the radar to determine if it would detect the missile. Now, if the missile has a flight time of 350 seconds (corresponding to speed 550 knots and distance 50 nautical miles) and its path intersects the detection area of the radar, we would have the model ask the missile and the radar 350 questions. To 348 of the questions the answer would be "no", the SSM is not detected, it will start answering "yes" when the SSM is detected and continue answering "yes" until the radar loses the missile. The time step simulation would mean asking for a large amount of redundant information with potential of a high computational cost.

In discrete event simulation, the simulated time is not advanced in predetermined steps. The idea is to determine exactly the simulated time a state variable will change and then advance the time directly to that point, thereby avoiding all the intermediate calculations that do not produce any usable information. Let us go back to the example of the SSM and the radar. Under the principals of discrete event simulation, the modeling of the detection process starts with the calculation of the point in time the radar will detect the SSM based on the velocity of the missile. If our missile again has a flight time of 350 seconds and we calculate that it will be detected after 144 seconds, say, time will be advanced 144 seconds in one leap, and then we change the necessary state variables.

Compared to time step simulation, discrete event simulation drastically reduces the number of times the program asks its components for their state variables. On the other hand, the mathematics needed to predict the time of future events might turn out to be more involved than what is needed to solve the same problem by using time step. Summing up, however, it seems clear that discrete event simulation is advantageous in modeling the problems in this thesis.

2. Stages in Model Development

a) Entity Level

The first stage in creating a simulation is to capture the relevant physics. Moving objects require certain attributes such as a location and a velocity. For some models it may be sufficiently accurate to assume that the objects change speed and direction instantaneously while other situations may suggest that a certain time with acceleration is necessary to change from one speed to another. Velocity changes in zero time are never the truth in reality, but using such a simplification in a model may be sufficiently accurate if the time used to change from one speed to another is negligible compared to the time with constant speed. If an intercontinental airliner underway is modeled, time to increase and reduce speed is very small compared to time with constant speed. Changing speed however is a main concern in a model of a car in inner city traffic. For the height above sea level on the other hand the requirements for accurate information will be the opposite, altitude is more important in a model of a car than in a model of a flying object. Other relevant physical relationships appropriate for a model might be curvature of the earth when obtaining objects positions, radar cross section of missiles we want to detect with a sensor or the number of shells available if we are modeling a gunnery system.

b) Interaction between Entities

The second stage in model development should be to capture the interaction between objects. For now, let object be a generic phrase used for the variety of instances of reality that we want to model. The tradeoff between a model that is detailed enough and no model at all, as indicated in the start of this section, takes critical effect in this stage. Deciding what to model is harder when it comes to interactions than when physical parameters are replicated. Allowing separate entities in nature to be represented by distinct entities in the model makes modeling the interaction between the two entities easier and more realistic.

Examples of entities from nature that are replicated in separate instances in the model in this thesis are of two types. First are the concrete entities like missiles of different types, radars and guns. Second are more complex systems, like missile batteries or combat information centers. The interaction we need in our model may be between two concrete entities like a missile and a radar, or between the missile battery and the missile the battery is launching. In either case it is a critical issue to model only the interaction needed to answer the question being asked. Once we have decided on the physical attributes to model, it is important that the interaction takes advantage of exactly all the features available. The interaction should not attempt to be more sophisticated than the actual features of the interactors allow.

For example: Suppose are we modeling a gun and have decided not to model the individual rounds fired, but rather use a rate of fire. When we model the interaction between the gun and its target at a later stage, this too should be based on the rate of fire. Modeling this interaction at the single shot level would mean presenting an unsubstantial high level of detail. On the other hand looking at the interaction between the gun and its target as a single event, not considering the time it takes to fire x rounds, would be an oversimplification of the model. In the latter case the effort put into modeling the rate of fire in the gun entity would most likely have no contribution to the model performance as a whole.

c) Human Interference

The third, and final, level of modeling should be interactions concerning the human being involved. Modeling at the "man-in-the loop" level is probably beyond the scope of a study of this kind, and it is not attempted. This kind of modeling would also make the subsequent data analysis considerably more difficult. Obviously the actions taken by persons are critical to the outcome of an anti-air warfare battle. If human action was not allowed in the simulation, the SSMs would not be fired, and the threatened ships would not act in self-defense. The way used around this is to assume that the result of all human performance is captured in standard operating procedures, tactics, and in letting the user of the model specify limitations on human performance. The tactics that are modeled are general; they do not rely on any specific Navy manual, but they bring into the model some generally accepted methods for conducting anti-air warfare. Further, as detailed in the previous chapter, time is a critical issue when a ship is under attack, self defense has to be activated before it is too late. The user of the package is given several options to select the distribution and magnitude of time used by personnel at various levels to make up their mind or to perform some manual action.

We have simplified the modeling of human interference to modeling the tactics or doctrines that involved personnel will use as guidelines for their efforts. The user is given the opportunity to specify the efficiency of involved personnel through setting parameters that dictates the time consumption associated with executing the tactics.

C. MODEL BALANCE

When building the computer models, capturing the basic concepts is relatively easy but the complexity of the model grows exponentially with increasing levels of realism and resolution, especially when dealing with details in interactions between several entities. Every model ultimately may face the point where tradeoffs between producing a good-enough model and not producing any model at all will have to be made.

When choosing where the cutoff should be it is vital that the model is well balanced to optimize the yield from effort invested.

A well-balanced model should be equally sophisticated in answering all its subproblems. Lets take the example of a model of a car in the city traffic. Say the model solves high order equations to determine the change in speed, and can predict arrival time at the next intersection with an accuracy of one hundredth of a second. Assume also that how the car turns is also modeled accurately, simulating its turn radius, maximum comfortable speed in the curve and so on. Now, if it turns out the final model cannot capture both speed change and acceleration at the same time, much of the effort is wasted. The model is unbalanced: It models two aspects separately to perfection, but the true nature of drivers to both use the brakes and the steering wheel simultaneously cannot be modeled. The model as one product would have been better if even a crude method of combining curving and acceleration had been implemented.

The answer to this dilemma is that the simulation model must be viewed as a single product. The different entities and the different areas of the model must be at the same level of sophistication. Only when all sub-problems are addressed at the same level of resolution is the complete product balanced.

This chapter has stated some basic rules that are generally accepted by most modelers. There should be nothing new in the previous sections, but as in many other fields it is easy to forget the first principles and they need to be clearly stated. In the next chapter we will go one step further and discuss the development of our model with the general rules mentioned above as background information.

III. FROM TACTICS TO MODEL

The first two chapters discussed the development and principals of naval weaponry and gave the broad and general guidelines that have been followed when the model in this thesis was designed. This chapter will take the reader from the overarching philosophies into the model at hand. We will first discuss the two hierarchies, one addressing the relationship between the pieces, objects, of software and one addressing the chain of command in the model. Object oriented design enables a complex system to be modeled by interacting components, closely imitating nature. In fact such programming makes it possible to discuss both the organizational structures of reality and of the components of the package under the same headline.

After describing the structure of the model we will address the infrastructure that is being used to make the model run.

A. OVERALL DESIGN AND BASIC PRINCIPLES FOR INTERACTION

In this model there are three separate areas addressed that should have the same degree of sophistication. The model, as a complete product, will be limited by the weakest of the three.

- The modeling of moving objects.
- The applications of sensors and the closely related dissemination of target information.
- The engagement cycle.

The overarching philosophy when designing the software package was to allow the individual classes of the computer code to represent distinct entities in nature. Based on the same arguments, the hierarchy of military units is also reflected in the hierarchy of Java classes as far as possible. The package is constructed with one chain of command for

the attacking side, those launching the surface-to-surface missiles, and one command structure for the ships trying to defend themselves against the attack.

To make the scenario unfold, however, we have to diverge from reality and construct a third participant, the neutral instances. The essence of the part played by such elements can be said to represent what is left to chance in the real world. That is, the neutral instances will introduce randomness to the model. We are modeling complex systems in anti-air-warfare, and many aspects of the AAW battle will be determined by chance even with the most sophisticated equipment employed to solve the challenges.

When it comes to computer code, however, the two sides of the engagement are not as separated. For example, when modeling missiles it is convenient to use the same backbone regulating physical attributes, like motion. After all, most of the features of SSMs and SAMs can be modelled with the same code by changing the parameters as necessary.

At this stage it is natural to orient the discussion in the direction of computer code. The software in the model is written in in Java™ version 1.1.7A, an object oriented programming language. From here on this text will use some terminology specific to the Java language and to the conventions used in its programming standards. An important aspect of those rules is however that the English meaning of the "Java word" and phrases are very similar. Hence a thorough knowledge of Java is not necessary to understand this thesis.

We will maintain the these standards in the following:

- Names of classes are written as `ClassName`, starting with a capital letter.
- The name of the class describes as far as possible, in one phrase, what the class is.
- An "instance" is one particular implementation of a class. There can be any number of implementations of each class.

- All the classes of a model are in packages. In our case all the classes in the model itself are in one package, the infrastructures are in separate packages.
- For classes from other packages than our primary, the names are given as `yourPackage.AndYourClass` and `myMuchBetterPackage.TheBestClass`.
- When referring to a particular method within a class the notion will be `someMethodInAClass()` starting with a lowercase letter and ending with a pair of parenthesis.
- If a method takes an argument the type of argument will be given if it will clarify the context: `someMethodInAnotherClass(TakingThisArgument)`.

1. Class Inheritance

Inheritance is a common phrase in object oriented design, it refers to letting one class inherit some or all of the attributes and methods of its parent class. Inheritance has been used extensively in the thesis code, described above as hierarchy of software. Normal terminology suggests calling the parent class the "superclass" and the child is referred to as the "extension". The following pattern of inheritance is established, the superclass given at the top:

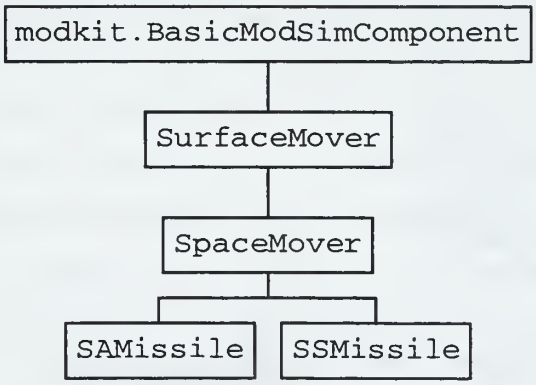

Figure 3.1. Inheritance for Classes Modeling Motion

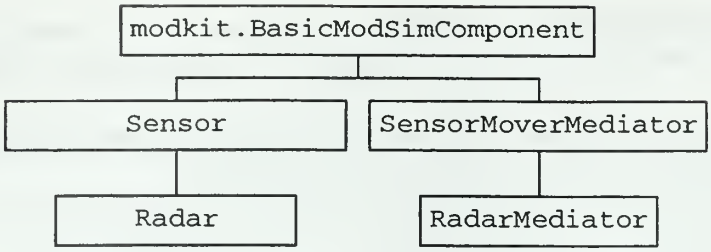

Figure 3.2. Inheritance for Classes Modeling Detection

The following classes represent weapon systems, sensors, organizations or neutral instances. They all extend `modkit.BasicModSimComponent` but are not extended themselves:

```
SAMBattery
SAMBatteryPassive
Gun
Tracker
FPB
SOFPB
CIC
OTC
GunMissileObserver
OutcomeObserver
```

The following classes extends `modkit.BasicModEvent` as part of the scheme for passing information and orders in the organizations:

```
DataLinkEvent
GunEvent
MissileEvent
MovedEvent
ObserverEvent
SensorEvent
TrackerEvent
```

2. Organization of the Defending Side

The organizational chart in Figure 3.3 shows the hierarchy between the different classes as they are cooperating in a chain of command in the computer model as well as in real life. The reader familiar with acronyms in naval warfare will recognize the

classnames as names used for entities in the actual organization of ships at sea. Notice that this organization differs considerably from the much simpler one described under the inheritance paragraph. The basic difference may be described as the above hierarchy working between classes, and the next hierarchy working for instances. The relationship in Figures 3.1 and 3.2 is a "is a" relationship, while the one in Figure 3.3 depicts a "has-a" relationship.

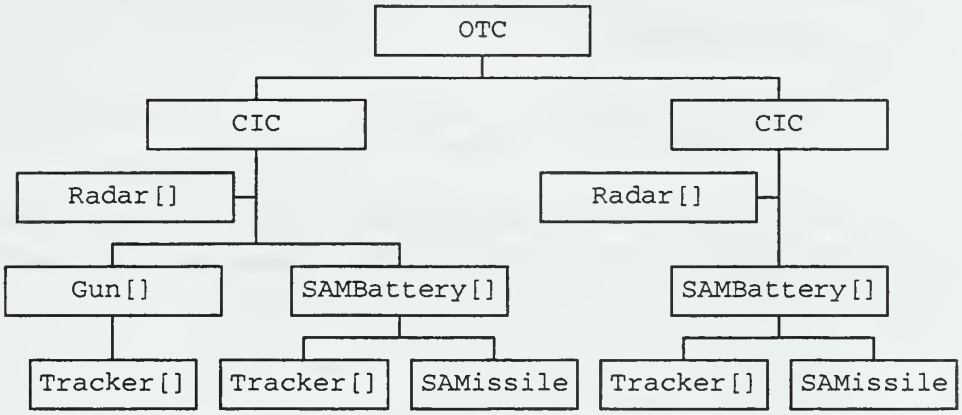

Figure 3.3. Organizational Chart Defending Side

The brackets [] in the above figures indicate that there may be an array of instances of the class. The OTC class represents the leadership of a group of ships at sea, the abbreviation OTC itself has the well defined meaning Officer in Tactical Command. The OTC has two of the duties normally associated with an officer in charge of a group of naval units. The primary task of the class is to do overall threat assessments, and to select one of its subordinate units to deal with each of the incoming threats. In an actual task group the OTC delegate Local Anti-Air-Warfare Commander (LAAWC) would exercise this responsibility. The second duty of OTC is to be in charge of the picture compilation in the force. The class will receive datalink events from all its subordinates, and relay them as necessary. When some tracks are lost the OTC will decide when to take the lost contact out of the tracking system.

The CIC instances are carrying the heavier workload in the package. In naval terminology the abbreviation means Combat Information Center. It points to the facilities in a ship where information is received and evaluated, decisions made and orders issued. The CIC does all of this at a detailed level. The principal responsibilities and working sequence of the class are as follows.

- The sensors will notify when a target is detected, CIC notes the position of the target and the time.
- CIC wraps the target into an instance of a utility class and sends it to the accompanying ships in the group, through the OTC.
- After some time has elapsed the class collects the new position of the target and based on the two observations and the time passed between the observations the velocity of the target is available.
- Based on the velocity a threat level is attached to the target. The new information is passed on datalink.
- If the contact is close enough to justify attack in self-defense, the attack process is initialized.
- If the target is not an immediate threat, wait for OTC orders.
- If an attack order is received, or the contact is immediately threatening, CIC starts an attack and reports back to the OTC when the action is complete.
- When a target is due for attack the CIC will select a SAMBattery, and when possible also a Gun, to counter the threat
- If a tracked target is lost (undetected) by one sensor the CIC will check its other sensors.
- If a target is no longer held on any of the CIC's sensors the other units are informed through the OTC.
- When one CIC is informed that one of the other CICs has lost track of a target it will take over as tracking unit for the force, given that is tracking the object with its own sensors.

Upon receiving an attack-order, the SAMBattery and Gun classes will check if the target is engageable, and if so select a Tracker to assign to the target. Once a Tracker reports back that it is successfully tracking the target, SAMs will be launched and the Guns will open fire at once or when the target is within maximum range.

When the attack sequence is complete or aborted, the CIC will signal this to its superior and continue with its next target in queue or wait for further orders from the OTC.java. The sequence is repeated until all targets are neutralized or they have hit their targets.

3. The Attacking Side

Compared to the defending side the picture of the classes on the attacking side is considerably simpler. Again the picture reflects the "has-a" relationship between instances.

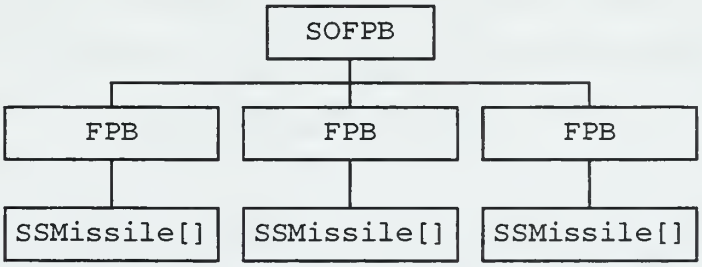

Figure 3.4. Organizational Chart Attacking Side

In NATO terminology the acronym FPB is for Fast Patrol Boat and SOFPB is for Senior Officer Fast Patrol Boats. The SOFPB will receive a point in time when the SSMs in the attack are supposed to have impact in their targets. It will convey this time to its subordinates, the FPB instances. Based on their own position, the route the SSMs are going to take, and the present position of the target, each FPB will determine when to launch the individual missile. The launch is done without any further notice. The most important part of an SSM attack that is not modeled is the phase of information gathering, when the officer in charge of the attack is building up her picture of the ships to attack.

This simplification is done under the assumption that the picture buildup phase will be equally successful under all choices of attack formations evaluated in this thesis.

4. Neutral Instances

As mentioned earlier we need a scheme for starting and monitoring the interaction between the attacking and the defending sides. The range of neutral classes is in charge of these procedures. They may be recognized by being the only classes with references directly to both sides in the simulation. The organizational structure of the neutral classes is shown in Table 3.1.

Attacking side	**Neutral instance**	**Defending side**
SOFPB	(The user, initial parameters)	OTC
SSMissile	SensorMoverMediator RadarMediator	Sensor Radar
SSMissile	OutcomeObserver	SAMissile
SSMissile	GunMissileObserver	Gun

Table 3.1. Class Organization

The first type of neutral instances created in the sequence is the instances connecting the incoming SSMs and the sensors of the ships under attack. When created, the `FPB.java` is given an array of `MediatorFactory` instances, and at launch time these are given a reference to the new SSM. The `MediatorFactory` then creates the necessary mediator instances, one for each pair of sensor and SSM. That is, if some ships have a total of seven radars and 20 SSMs are fired in the attack there will be a total of 140 mediator instances.

The `SensorMoverMediator` is the superclass to `RadarMediator`. The first class is a pure cookie cutter mediator, but the extension is a more sophisticated and realistic mediator. A cookie cutter sensor automatically detects a target as soon as it enters the maximum range, and is a simplification from reality where we will observe a

time delay between entering maximum range and actual detection. Both the mediator classes have a single responsibility. Based on data about the sensor and the target they will decide when, if at all, the sensor detects the target. Once the target is detected the mediator will calculate the time it will leave the detection envelope of the sensor. Two key assumptions are made to allow the mediators in this thesis to work properly.

- The target should maintain constant velocity.
- Time spent changing velocity is small compared with time spent with constant velocity.

Implicit in the first assumption is the fact that the mediators are not capable of predicting times for detection for targets that are accelerating or changing course. The assumption does not by any means exclude the targets from maneuvering; it just emphasizes that every time a target has changed its velocity the old entry or exit times will have to be discarded and new ones calculated. This assumption has only small consequences, as recalculating the entry- and exit-times is computationally affordable. The second assumption follows the first one closely. By default the movers notifiy the mediators only when they have completed a change of speed, course or altitude. For example, the arrival times calculated on the old course is changed abruptly when the mover is steady on its next course. Hence the times existing in the system during the maneuver are not correct.

The second assumption is made also for two reasons. The alternative would imply drawing to much attention to a narrow subset of the total problem, and jeopardize the balance of the model. The other option would be to provide the mediator with information on how long a turn or acceleration would go on and allow it to calculate the entry and exit times based on the associated higher order equations of motion. The danger with this is that the final product could outsmart the reality; that is the simulation would be capable of predicting into the future with unrealistic high degree of precision. The most important argument in justifying the second assumption, however, is the fact that

SSMs generally move with steady or near steady velocity in most of their flights. Some systems do weaving on their course to offset countering weapons in their final seconds, but these maneuvers are done at close range when detection is an issue.

After a detection, the defending forces will eventually have to engage the incoming targets, again requiring neutral instances to act as referees for the encounters between weapons from the ships and the Surface-to-Surface missiles and between the SSMs and the ships. The `OutcomeObserver` class is responsible for the interaction between Surface-to-Air missiles and SAMs and between the SSMs and their target ships. The `GunMissileObserver` is umpiring engagements done by the `Gun` instances.

B. INFRASTRUCTURE

The computer model in this study uses the infrastructure of two separate software packages, Simkit and Modkit. Simkit is designed and maintained by the Department of Operations Research (OR) at the Naval Postgraduate School (NPS). It is made available to public through the Internet (http//web.nps.navy.mil/~ahbuss). Simkit has been the core of the classes in simulation taught at NPS. Modkit is a software package constructed by Major A. Arntzen, Royal Norwegian Air Force in his thesis (Software Components for Air Defense Planning, 1998).

Simkit is a well proven and frequently used package, while Modkit is less well known. Because of the limited experience in use of Modkit, this package is given a more thorough discussion than Simkit.

1. Simkit

Simkit is the tool that makes the model into a discrete event simulation. The Simkit packages have all the necessary components for supporting a full-scale simulation alone, but some of the features offered by Simkit have been abandoned in favor of similar possibilities available in the Modkit package, to be discussed in the next section. We are using Simkit for the following support:

- Event scheduling, time keeping
- Event executing
- Producing Standard Uniform random numbers
- Collection of statistics
- Debugging

The two first items are by far the most involved, and they are closely connected. Under the rules of Simkit it is the responsibility of the extensions of `simkit.SimEntityBase` to calculate the time for the next event. When this is done the event scheduler of Simkit is called and the particular event is placed on the event list. For example: A ship has just completed a turn and is travelling with a speed of 15 knots towards its destination A. When it arrives at A we want to give it another position, B, to navigate to, and we will decide on this new destination in a method called `doArriveDestination()`. Consequently we need this method to be called at the moment the ship arrives at A. When the turn is completed at A then, and the ship is heading for its new destination our responsibility is to calculate when the ship will arrive at B. If the distance is 7.5 nautical miles it is easy to determine the delay to arrival being 30 minutes. We should then tell Simkit to invoke the method `doArriveDestination()` after a delay of 30 simulated minutes. When the arrival event comes to the top of the event list Simkit will find and invoke the correct method.

The event list executes the scheduled events in the sequence of their simulated time so that all events are executed chronologically. Also, the event list that is created by the time master may be displayed for efficient debugging purposes.

Simkit has sophisticated measures for producing random variates, but we are only using the core methods, which returns random uniform numbers in the interval (0,1). For the model at hand the necessary variables are produced locally based on transforming the numbers produced by Simkit. The statistics package of Simkit is simple, but it provides a convenient way of obtaining the most basic statistical summaries.

2. Modkit

The principal philosophy behind Modkit is that modularization of computer programs will make them more effective, in the sense of making them easier to reuse and to change. To accomplish the modularization Modkit introduces a scheme of distributing information, through event handling, and dissemination of attributes from one class to another, by property dispatching.

As pointed out above our goal should be to simulate separate things in separate classes and the interaction between them should also follow the rules that exist in nature. In our case issuing and executing orders are essential parts of the interaction we need to model. In particular, the way orders are given down the chain of command and information is passed horizontally and upwards in the hierarchy should be reflected in the interactions of entities in the model.

a) Event Dispatching

Imitating the flow of information within a ship, or within a group of ships, is better implemented by a message passing approach than the traditional way of invoking methods. Here information may be passed in a message and it should not be critical to the originator who gets it first or if someone receives it not intentionally on the list of recipients. Following these principals, when the `Tracker` has gained contact it sends out a message to those on the address list, not concerned if other entities than the one that issued the tracking order are also informed. Using parallel reasoning when a ship is done with a SAM-attack it sends out a message with information of the outcome to all of those listening.

The Modkit way of generating and handling events in the simulation is not unique, but after building the present model it may be concluded that the way it is constructed in this infrastructure is useful and efficient. Event dispatching is used extensively and the features of the event procedures of Modkit very closely resemble information flow in an organization. Events are used for general distribution of

information, but particularly to pass information back to the superior when the result of executing an order is clear.

Event dispatching is designed to happen in zero time because issuing and receiving of events are done independently of the simulated time and the event list. In a computer simulation however, basically only one thing can be done at a time and then by default, the dissemination of a message will be to one recipient at a time. If we want action to be taken by an entity upon the notification of an event we should put a delay on this action if it is not desired to have it happen "nested" inside the event dispatching. As an example: Ship A intends to send a message to ships B and C. Receiving this message makes B take action of some sort, which again may result in new messages being sent from B. If sending of the message from B is not executed through the event list (delayed) it will be done before the original message from A is received by C. In some situations nesting of events may be acceptable while others may require delaying subsequent events. The event handling scheme is a versatile tool that is used with success, ease and realism.

b) Property Dispatching

While event handling is used when possible, property dispatching is used when necessary. The event dispatching needs both an originator of a message and one or more recipients to make sense, but property dispatching is a one sided issue. It is also natural to classify it into a "passive" where we allow one instance adopt some of the features of another without using inheritance, and an "active" part we invoke a dispatched method in an instance.

(1) Passive property dispatching. The passive part of the property features is used to give a few but very powerful solutions to the challenges in the model. The best example might be the `Radar` instances that are given the properties of the `SurfaceMover` instances. To solve any problem of detection it is necessary to know the position of the sensor and the object it is looking for, but the `Radar` class has no means of holding or calculating any information of its location. Instead it is given the

properties of the `SurfaceMover` class. This class holds any feature regarding motion in two dimensions presented in the model. By doing it this way someone may ask the `Radar` for its position and be answered correctly, and the answer is really the position of the `SurfaceMover` (ship) that carries the sensor. By doing nothing to the source of the properties, hence the categorization to passive, the sensors and other objects can be given attributes that otherwise would have meant much larger and complicated classes. The procedures described here are commonly referred to as "delegation".

(2) Active property dispatching. During the design of the package, the active half of the property dispatching was never used. When modeling something that is supposed to resemble a military system or command structure it does not make sense to use anything but direct invocation of a method when the desire is to allude an order. Use of active property dispatching would contradict our goal of letting instances and procedures in nature be represented by similar instances and procedures in the code.

Following these arguments, when the `OTC` instance decides to order a subordinate ship to execute a SAM-attack it is done by invoking the method `engageTarget()` in precisely the instance of the `CIC` class that has been judged to have the best possibilities of success. Similarly when a `SAMBattery` has been given an order to launch surface-to-air missiles it will invoke the `startTracking()` method in an instance of the `Tracker` class.

c) Modkit Coordinate System

Modkit has a package, `Modkit.modutil.spatial`, providing geometric methods in Cartesian coordinate systems. The classes that have been used in the model of this thesis are `Coor3D` and `Coor4D`. The two classes provides three dimensional coordinate systems, and methods to compute vector algebra in x,y,z space. The fourth dimension in the `Coor4D` position vector is essentially ignored when algebra are performed, for example when the distance between two points are computed the

Pythagorean theorem is used only on the first three positions, the x-, y-, and z-coordinates.

The classes are used for positioning in this project. The convention that has been maintained is to use three-dimensional vectors, from the class `Coor3D` for velocities and four dimensional, `Coor4D`, for positions. The fourth dimension in the `Coor4D` has been used as a placeholder for the time at that position. This is convenient, but other usage of this is also possible.

We have now discussed both the overarching philosophies for modeling, the organization of the `kitchen` package and the infrastructure that has been used. Our next step will be to explore some of the non-trivial algorithms and assumptions built into the model in the software.

IV. THE KITCHEN PACKAGE[1]

This chapter will discuss the model constructed for this thesis in detail. The package is written in Java™ version 1.1.7A. We will address the structure of the package design; the assumptions and approximations made to model nature. Some of the more important classes will be discussed in details and examples of code will be shown. Resulting from the increased level of detail it is necessary also to go deeper into some specific Java procedures. For a general description of the Java language see the list of references. (The Java Language Specification, 1996) and (The Java Class libraries, 1998).

A. DEFINITIONS AND GENERAL ASSUMPTIONS

1. Units of Measure

The units of choice are a result of desire for convenience and familiarity rather than demand for high accuracy or adhering to International Standards (SI). Consequently the nautical measures for distances and speeds are adopted. To avoid unnecessary conversions however the maritime preference for degrees to measure directions has been abandoned to take direct advantage of the trigonometric methods of the `java.Math` class, using radians.

- All distances are measured in nautical miles.
- 1 nautical mile = 6000 feet.
- Time is measured in seconds.

---

[1] The name "kitchen" could come from the NATO nickname for a huge Soviet - Russian anti shipping missile or it could come as protest to the scores of acronyms floating around in military language. The truth is however that the author was accused of spending too much of his time by the computer and too little time in any other room in the house. Hence, the package was named `kitchen` and the execution class was named `Table`. Lots of time was thereafter spent deep inside the kitchen.

- True bearings measured clockwise in radians, range from 0 to 2π, $0 = 2\pi$ = true north.
- Relative bearings measured clockwise in degrees, range from 0 to 360, 0 = 360 = front (bow) of unit.
- Probability of hit for a weapon (dimensionless) ranging from 0.0 to 1.0.

All units are in double precision, except for relative bearings, which are integer. The units derived from the basic ones above:

- Speed is measured in knots, nautical miles per hour (3600 seconds).
- Velocity is a three-dimensional vector of speeds, in x,y and z direction respectively.
- Acceleration is knots per second.
- Turn rate is radians per second.
- Rate of climb or dive for flying objects is in knots.
- A position is a four-dimensional vector with distances from an origin in x,y,z space in the first three positions. If the fourth position is nonzero it will hold the time at that position.

2. Positioning

The positioning system is made as simple as possible. When the model is initialized all the participants are given start positions in a Cartesian grid system. All the objects are mere points, none of them have any size. The positive y-axis is pointing north, the positive x-axis pointing east. Positive z-values represent height above the sealevel. As a consequence of this simplification we will introduce a certain amount of inaccuracy due to the fact that the earth is not flat. This will however not invalidate the model as long as the distance between the various objects are of the same magnitude as the range of surface-to-surface missiles of today, say 150 nautical miles or less. Although this divergence from nature is acceptable when it comes to pure positioning accuracy,

there is another very important issue of the curvature of the earth, the range to the horizon. Assuming that the earth is flat would also imply that we would have no horizon and that any object would be theoretically visible at any distance. This issue about the horizon is however played despite the simplification above, and objects will be both above and below the horizon.

Further, a consequence of the flat surface assumption is that no landmasses will be modeled. This might be a serious breach with the actual situation when SSM attacks are launched from Fast Patrol Boats or aircraft in the skerries where some missiles possibly would be masked by landmasses for parts of their flight. The workaround is however simple, and the alternative would mean a significantly more complicated model. To imitate SSMs doing some of their flight toward the target area hidden by islands or mountains the firing position should be placed where it would become detectable. Ranges and times should also be adjusted accordingly.

3. Success Criteria

The user must specify the hit probabilities of the various weapons. They might be fixed numbers, or a function of the engagement range. The values are assumed to capture only the probability of the weapon hitting the object at which it is aiming. That is, the probability of an unsuccessful missile firing, the missile failing to leave the launcher, is assumed to not be covered by the hit probability. If the expected number of missiles that will not work properly is significant, the user should reduce the number of weapons available to compensate for this in the kitchen package. The same goes for the simulating of gunnery systems, if it is expected that malfunctions will hamper the performance of a gun this should be reflected in the rounds available for each engagement or by setting the rate of fire at a realistic level. In short the probabilities of hit specified by the user should be the probability of hit given that all associated systems are working successfully.

It is assumed that one hit with a Surface-to-Air missile or one gunnery round is sufficient to make an incoming Surface-to-Surface Missile unsuccessful. This will

probably resemble nature closely, as SSMs are not known to be able to survive hits from neither guns nor missiles.

B. MOVING OBJECTS

The reason for modeling moving objects in the package is the necessity to observe where, when and how the various objets interact. Without motion and changes of motion it would be nearly impossible to model an anti-air-warfare scenario. The critical issues are range and time of detection of incoming threats, opening and closing of weapon arcs on ships changing course, and finally the physics of bringing a self-defense weapon up against the threat. It was not feasible to take this to the highest level when it comes to two objects actually hitting each other. That is, with the level of accuracy motion is modeled the distance between a SAM and a SSM can be larger that what is the actual "kill radius" of the SAM warhead and the model still count the incident as a hit.

All the algorithms for motion are concentrated in the classes `SurfaceMover` and `SpaceMover`. As the names should indicate, the two-dimensional movements are captured in the first while the latter class holds methods for three-dimensional motion. The `SpaceMover` is an extension of the `SurfaceMover` and by inheritance any method in the super class is normally available in the extension. In the simulation the ships will be instances of the two-dimensional class and the missiles will be of classes inheriting from the `SpaceMover`.

Although missiles are flying objects, the differences in the height-component of their positions are small compared to the differences in the x- and y- component. Typically a SSM will operate in the altitudes 0 to 100m say, while an extremely short firing range would be 7000m. Also, for both SSMs and SAMs, the time spent changing from one altitude to another is short compared to time with constant height or near zero climb or dive rate. With this ratio in mind, it was decided to make the algorithms for up-and-down motion far less detailed than those capturing the horizontal components. The

model would be better balanced by focusing most of the resources available for modeling motion into the `SurfaceMover` class.

1. Two Dimensional Motion

The normal procedure in the `SurfaceMover` class is that changes of speed occur at a specified rate, and changes in heading occur at a specified turn rate. The rate of speed change is constant for both acceleration and retardation. That is, acceleration from 5 to 7 knots will take the same time as acceleration from 12 to 14 or retardation from 19 to 17.

The turn rate is a function of the speed of the instance. Turn rate is set at a small value for speed zero, then it increases linearly to a maximum at about half of the maximum speed of the object. The rate stays at maximum up to 80% of full speed of the objects, then it drops slightly for higher speeds. These simplifications are assumed to be more than accurate enough for the problem we are analyzing, additional assumptions are required objects with simultaneous turn and acceleration. Modeling this accurately is an involved process, so some heuristics are used. When a `SurfaceMover` instance is about to change both speed and course, the speed is set, without acceleration, at some value between the current speed and the ordered speed, and this speed is used to determine the turn rate. When the turn is over, regular acceleration is done to achieve the ordered speed. For the purpose of this thesis, this is sufficiently accurate; but for a simulation primarily looking at the motion itself, it is probably not accurate enough, especially if the speed is zero at the time when both acceleration and turn begins.

The class has several navigation variations, but the principles are much the same. The instance will always have a start position and a position to navigate towards, called the current destination. It has discrete event methods to start the object the first time, and the methods `startAccelerate()`, `stopAcellerate()`, `startTurn()` and `stopTurn()` to change its state variables. When one of the methods `stopAcellerate()` or `stopTurn()` is invoked the next method invoked, would be `arriveCurrentDestination()`. At the time the latter method is invoked, the

mover should have been given a new position to move to and the necessary maneuvers to get to this position are executed.

To get from point A to point B, the mover can be given the arrival time and adjust its speed accordingly or it can be given a specific speed. Also the mover may be given a number of positions to navigate trough on its way to the destination, called waypoints. Because of the turn radius, which is a function of turn rate and speed, compensation is done for the transfer on the "old" course to actually hit the desired point on the new course. Also, the class holds two higher order methods for navigation that will be discussed next in more detail.

a) Intercepting

The `intercept()` method may be used when the system we want to model has a capability of predicting future positions of a target, and is used every time a SSM is launched, and under certain conditions also for SAMs. The arguments to the method are the present position of the unit desired to intercept, its velocity, the speed this unit is going to use, and whether the mover should first traverse any waypoints or intercept immediately. The assumption is that the target will maintain constant velocity, and the method will predict the intercept position. If the target changes its velocity before the actual interception occurs, the method would be called again if interception is still desired.

When the necessary second order equation is solved, it uses the ordered speed of the interceptor as a parameter. If the interceptor needs to turn, and maybe also accelerate, to reach its interception velocity, there will be inaccuracies in the predicted position proportional to the time used to achieve the correct course and speed. To eliminate this inaccuracy in the model would require invoking this method one or more times after the initial change of velocity. The missile classes that inherit from this class update their intercept position at least once after the initial predictions are computed before launch. We will revisit this method below when we discuss the missiles.

b) Tail Chasing

The second method to achieve interception between a mover and a target is the `tailChase()` method, to be used by objects not capable of predicting the position of the actual interception. Here the movers will always move directly toward the present position of the target. The picture could be a dog chasing a ball; the animal will not be smart enough to predict the future position of the ball, and hence it will point its nose at the ball and follow it.

The method takes the same arguments as the `intercept()` plus a value for how often updates should be done, the update interval. In contrasting with the `intercept()` method, this method does not use a closed form solution. The principle is to create a large number of waypoints for the mover to follow on its way to the point of intersection. The first waypoint will be placed in the direction of the present position of the target with the distance down the bearing equal to distance traveled by the mover in one update interval. To fix the second waypoint, the bearing between the first point and the position of the target after one update interval is established. The second waypoint is placed down this bearing the same distance as to the position of the first. The same procedure is continued until the position of the next waypoint coincides with the assessed position of the target.

Obviously the `tailChase()` method is a computationally expensive, and some regressing to the method `intercept()` is done when feasible. For example, when the interceptor and the mover are approaching each other head on, the trajectory of the mover will be identical for both methods. Significant savings in computations are made by doing some initial checking of the resulting paths from the methods, and if the difference is acceptable the `intercept()` method is used.

2. Three Dimensional Motion

As mentioned above, the procedures in this class are left at a more basic level than the ones in the two dimensional case. Changing of altitude takes place with constant rate,

the units are knots, and the rate is the same for going up and down. Further the z-component of the velocity is allowed to change abruptly from zero to the specified rate; no acceleration is used with this state variable. Change of altitude is done through the methods `changeAltitude()` and `levelOut()` and may be done independently of changing of the state variables in the superclass.

C. SENSORS AND TARGET DETECTION

At initialization of the program the ships are given one or more sensors. The sensors are given the ship they are placed on as a source of properties to make it possible to compute distances to the various targets based on the position of the ship. When the SSMs are launched, the neutral mediators between the SSM and the sensor are created. The mediator listens to the events generated by the SSM and, upon hearing an event that says that the mover is at a new velocity, the mediator calculates the times for entry and exit of the coverage area of the sensor. An enter range event is put on the event list to be executed at the time for entering sensor range.

Although considerable improvements have made sensors more efficient over the last 15 years or so, the radar systems are still the dominating sensors in anti-air-warfare. Because of this, the following discussion will be focused on the modeling of radar systems.

1. The Constant Rate Detection Process

When a target enters the range of a sensor it is generally not detected immediately, and the time delay from earliest possible time of detection to actual detection is a random variable. The simulation of this variable is done rather carefully in the `kitchen` package. The following derivation is closely linked to the derivation of a general Poisson process:

> We assume that at every time u there is a rate of detection $\gamma(u)$ with the properties that

a) $\Delta\gamma(u)$ is the probability of detection in a small interval of time Δ that includes u.

b) the events of detection in non-overlapping time intervals are all independent.

Assumptions a) and b) determine the probability $p(t)$ that detection will occur somewhere in the interval $[0,t)$. To show this, let $q(t) = 1 - p(t)$.

Then

$$q(t+\Delta) = q(t)(1-\gamma(u)\,\Delta).$$

since the events of no detection over the intervals $[0,t)$ and $[t, t+\Delta)$ are independent. Therefore

$$\frac{q(t+\Delta)-q(t)}{\Delta} = -q(t)\gamma(t).$$

Taking the limit as Δ approaches 0, we obtain

$$\frac{d}{dt}q(t) = -q(t)\gamma(t).$$

The solution of this differential equation is

$$q(t) = e^{-n(t)} \quad \text{where} \quad n(t) = \int_0^t \gamma(u)\,d(u). \quad (4.1)$$

(Washburn, 1996, Chapter 2)

The interpretation of $n(t)$ is the mean number of detections in the interval $[0, t)$. Obviously the mean number of detections will depend on the length of the time interval but also the function γ.

With most sensors, detection will not occur until a certain amount of energy has been transported from the target to the sensor. For an active sensor this energy is first sent from the sensor, then reflected by the target and finally received by the sensor. With passive sensors the device is trying to receive any energy emitted from the target. These very basic principles hold for sensors using sound, like passive and active sonar, optical sensors like thermal devices and sensors operating in the traditional electromagnetic field

like radar. Further, common for essentially all of them is that the probability of detection goes down as the distance between the target and the sensor increases. This is a consequence of the loss of energy due to spreading, absorption and scattering in the atmosphere.

2. The Nonhomogenous Detection Process

Going back to the rate of detection $\gamma(u)$, a model that intends to capture the detection process closely will have to take into account that $\gamma(u)$ is not constant, but varies with the range. The process we want to model is a non-homogenous Poisson process, a Poisson process with a time varying rate. We can easily translate the detection rate as a function of range to a function of time given that we know the position of the target. At all times in the simulation, this information is available in the neutral mediators and the computations are done there.

We will write the distance-dependent probability of nondetection in the time interval $[0,t)$ as

$$1 - F(t) = \exp\left(-\int_0^t \frac{\alpha}{[R_t(x)]^n} dx \right) . \qquad (4.2)$$

Here the detection rate $\gamma(u)$ has been replaced by a rate depending on α, a constant capturing the target's detectability or "stealth-factor" and $R(x)$, the target range at time x. For radar systems α will be proportional to the radar cross section of the target and normally $n = 4$, as the returned energy to the radar antenna is proportional to the inverse fourth power of the range assuming inverse square law spreading in both directions.

Our goal is to create a random variable capturing the amount of time that elapses between the moment the target enters the range of the sensor and the moment detection actually occurs. We will use (4.2) and the inverse transformation method (Law & Kelton, 1991, p 465), to generate the random times of initial detection.

$F(t)$ is a cdf, or possibly it is a defective cdf. In either case F evaluated at any time $t > 0$ will be a number representing the probability of having a detection in the interval $[0,t)$.

$$p = F(t) = P(T \le t), t \ge 0, p \in [0,1]. \tag{4.3}$$

The inverse transform method is based on the fact that if V~Uniform[0,1], then $F^{-1}(V)$ is a random variable with cdf F. We will follow these steps:

1. Draw a random number V, $V \sim Uniform[0,1]$.
2. Find t such that $1 - \exp(-\int_0^t \gamma(\tau) d\tau) = V$.

   Equivalently, we numerically integrate the detection rate $\gamma(\tau)$ until

$$\int_0^t \gamma(\tau) d\tau = -\ln(V). \tag{4.4}$$

3. Substituting the detection rate derived in (4.2) into (4.4) gives the final equation

$$-\ln(V) \le \int_0^t \alpha / [R_t(x)]^n dx \tag{4.5}$$

Notice the inequality sign of (4.5). When doing the numerical integration we will generally not achieve exact equality. The smallest value of t that makes the integrand equa or exceed the threshold will be returned as the random time to detection.

As mentioned above $F(t)$ may be a defective cumulative distribution function.

That is, $$\lim_{t \to \infty} F(t) < 1. \tag{4.6}$$

The practical implication is that there is a positive probability that detection of the target never occurs, no matter how long the sensor is looking for it. Consequently we need an upper bound on how large we allow t to become in the numerical integration. Because the `kitchen` package is concerned with moving objects, the upper limit for t in (4.5) is the time the object leaves the range of the sensor. So if the threshold is not exceeded by the time the moving instance leaves the range of the sensor the algorithm will return

"Infinity" (`Double.POSITIVE_INFINITY`) which in turn will be interpreted as no detection occurring.

3. The Sensor – Mover Geometry

Before we can make the detection equation work, we will need a useful expression for $R(t)$, the distance between the sensor and the target as a function of time, and we will need the possible ranges of the sensor, R_{max}.

a) System Range

First we will discuss R_{max}, the outer bound for positive probability of detection. There are two factors that together determine the maximum range of a sensor, they are the limit built into the system by design, and it is the sensor's capability to "see" around obstacles. For radar the limit for detection range is typically given as a function of the pulse length and the pulse repetition interval. Most active range finders use the time interval between releasing energy and receiving the echo pulse multiplied by the speed of the pulse, and the maximum system range will be given by the time the system is in receiving modus before sending the subsequent pulse. This holds generally for radar, sonar and laser range finders. For a passive system there is normally not a discrete outer limit for detection as described here.

b) Horizon Effects

The other factor determining R_{max} for the radar is the distance to the horizon. Electromagnetic energy transmitted from a radar will not penetrate terrain or water significantly more than visual light. Radio signals used by radar typically have a wavelength in the range of 3 to 10 cm and consequently they will be marginally more likely to "curve" over the horizon than visual light. This curving, refraction, depends on the wavelength of the radar; the lower the frequency; the more refraction and greater range. Also the distance to the horizon is a function of the height of the point of view, for a radar the antenna height.

$$R = 1.22\left(\sqrt{h_a} + \sqrt{h_t}\right) \tag{4.7}$$

(Bowditch, 1977, p 946)

In Equation (4.7) R is the range to the horizon in nautical miles, h_a and h_t are the heights above sea level, measured in feet, for the antenna and the target respectively. The factor 1.22 in (4.7) corresponds to a radar with wavelength 3cm and frequency 9GHz, common parameters for navigational radar. Other sources give the factor as 1.23, presumably corresponding to a radar with lower frequency. We will use 1.22 throughout this thesis.

To resemble nature, we will have to determine the maximum range due to the horizon effect for every target sensor pair and compare it with the system range. The R_{max} to use for calculations will always be the shorter of the two. For targets at large altitudes normally the system range will be the limit, while for low flying objects or objects on the surface the horizon effect will be dominating.

c) Range as a Function of Time

After deriving the maximum range the second step is developing $R(t)$, the range as a function of time. Assume first that at time t_0 the target is in position (x_0, y_0) and that this position is outside the possible range of the sensor. Let (v_x, v_y) be the velocity vector of the target. When this problem is solved in the `kitchen` package a temporary positioning system is created, placing the sensor at the origin. The vectors (x_0, y_0) and (v_x, v_y) are consequently relative to the sensor. At time t the position of the target relative to the sensor will be given by

$$(x_t, y_t) = (x_0 + t\,v_x, y_0 + t\,v_y) \tag{4.8}$$

Simultaneously, the distance between the target and the sensor as a function of time $R(t)$ are

$R(t) = \sqrt{\left(x_t^2 + y_t^2\right)}$ which expands to

$$R(t) = \sqrt{\left(x_0^2 + 2x_0 t v_x + t^2 v_x^2\right) + \left(y_0^2 + 2y_0 t v_y + t^2 v_y^2\right)} \tag{4.9}$$

$$R(t) = \sqrt{t^2\left(v_x^2 + v_y^2\right) + 2t(x_0 v_x + y_0 v_y) + \left(x_0^2 + y_0^2\right)} \tag{4.10}$$

We recognize the terms in (4.10) as follows

$v_x^2 + v_y^2$	The relative speed squared
$x_0 v_x + y_0 v_y$	The inner product of the original position vector and the velocityvector
$x_0^2 + y_0^2$	The position vector squared, equivalent to the distance squared

To find the time the first detection is possible, we equate R_{max} with (4.10), and solve for t. The solution will either be two, one or no real numbers. No real solution means there are no real roots to the quadratic equation. The interpretation of these possible solutions is as follows:

- With two real solutions we have the entry and exit times of the detection disc. Negative numbers imply that the event of entry and/or exit has already occurred.
- Exactly one real root implies that the path of the target is tangential to the detection area. With double precision datatypes representing both time and distance this is almost a zero probability event
- No real roots imply that the path of the mover does not intersect the detection radius.

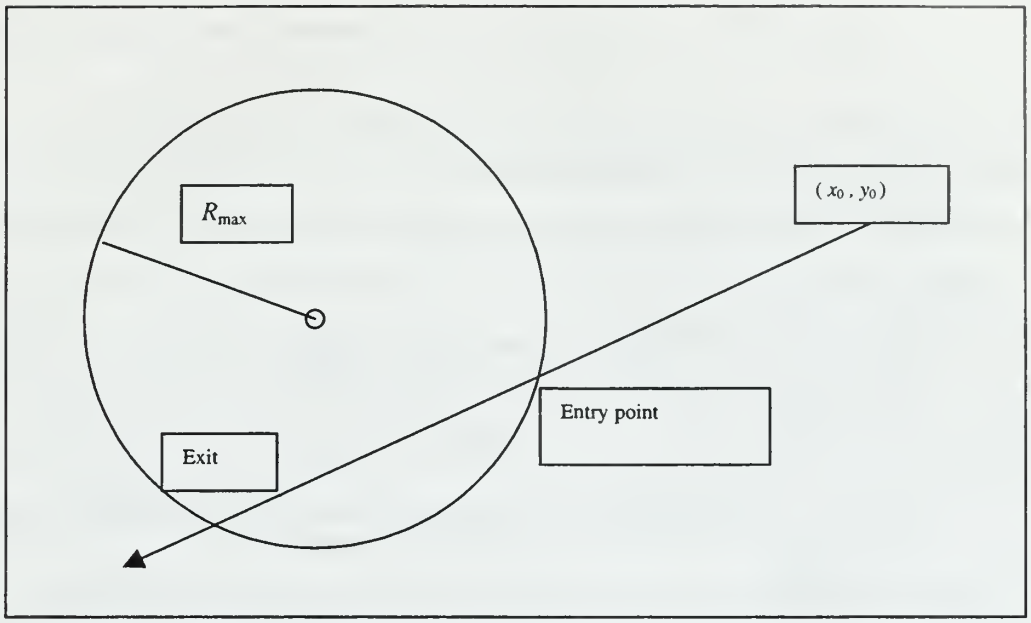

Figure 4.1. The Mover-Sensor Geometry

After establishing the elapsed time between the entry and the exit points, we may return to (4.3) and do our numerical integration. The following procedure to determine time of detection has been established and is used in the `RadarMediator` class.

- To operate the class we need to specify the parameters α, the "stealth factor" of the object and n. Increasing α means quicker detection and n should be set according to the sensor at hand. For an active radar n should generally be four. Increasing n with fixed α will lead to later detection and reduce the variance of the time to detection.

- The class will calculate the entry and exit times when it is informed that the mover has reached a steady velocity.

- The threshold is established, according to the left hand side of (4.5).

- Numerical integration is started with $t = 0$, corresponding to the point in time the mover crosses the maximum detection radius of the sensor.

- Integration by the trapezoid rule is done until the accumulated mean number of detects is equal to or larger than the threshold. The corresponding value of t is returned as time for detection.

- If the threshold is not met before the time in the integration exceeds the exit time, no detection will occur.

```
public double getDelayToDet(double lastPossibleTime){
   double threshold = - Math.log(myStream.uniform());
   double accumulated = 0.0;
   if(lastPossibleTime/deltaT > (double)maxIterations){
      deltaT = lastPossibleTime/(double)maxIterations;
   }
   double t1 = 0.0;
   double t2 = deltaT;
   if(method.equals("InverseSquared")){
      accumulated += deltaT * (alpha/getDist2AtTime(t1) + alpha/getDist2AtTime(t2)) * 0.5;
      while(accumulated < threshold && t2 <= lastPossibleTime){
         t1 += deltaT;
         t2 += deltaT;
         accumulated += deltaT * (alpha/getDist2AtTime(t1) + alpha/getDist2AtTime(t2)) * 0.5;
      }
   }
   if(method.equals("InverseQuadratic")){
      accumulated += deltaT * (alpha/getDist4AtTime(t1) + alpha/getDist4AtTime(t2)) * 0.5;
      while(accumulated < threshold && t2 <= lastPossibleTime){
         t1 += deltaT;
         t2 += deltaT;
         accumulated += deltaT * (alpha/getDist4AtTime(t1) + alpha/getDist4AtTime(t2)) * 0.5;
      }
   }
   if(t2 <= lastPossibleTime){
      return 3600.0 * ((t2-t1)*myStream.uniform() + t1);
   }
   else{
      return Double.POSITIVE_INFINITY;
   }
```

Table 4.1. Edited Code for Creating Random Variable for Time to Detect

Observe in the code in Table 4.1 that the method takes the argument `lastPossibleTime`, representing the time the mover exits the detection disc. The `RadarMediator` class has a default delta time to use in the integration as well as a max number of iterations to make. The user may set both these parameters. The call to the methods `getDistance2AtTime()` and `getDistance4AtTime()` will return the distance between the mover and the sensor at the point in time given as argument. The variable `posDotVel` is the inner product of the position and the velocity vectors.

```
private double getDist2AtTime(double t){
      return relativeSpeed*relativeSpeed*t*t + 2*t*posDotVel + distance*distance;
}

private double getDist4AtTime(double t){
      return getDist2AtTime(t)*getDist2AtTime(t);
}
```

Table 4.2. Methods for Providing the Distance Between the Sensor and the Mover

d) Detection

The procedures described above produce random numbers to represent the time of entry of the detection area and subsequently the actual time of detection. The algorithms are in RadarMediator and its superclass SensorMoverMediator. When the mediators have calculated the time for detection they will schedule to inform "their" sensor at the actual time. When this event reaches the top of the event list, the sensor will be informed by invocation of the method setDetection(). The sensor immediately adds the target to its list of tracked objects and sends a SensorEvent to its combat- and information center. The detection process for this target is now concluded.

D. PICTURE COMPILATION AND INFORMATION DISTRIBUTION

When the target has been processed by the neutral instances, it is received by the `CIC` instance that owns the sensor. This `CIC` will now start gathering information about the target. Two classes comprise the core mean for information collection and dissemination in the `kitchen` package, `Contact` and `ContactManager`. Every new target is wrapped into an instance of the `Contact` once it is detected. When the ship that detected the new target sends the new information to the other ships, these units too will create new instances of the same class based on the information provided by the first ship, even if they do not hold the target on their own sensors. As the situation develops, all `CIC` instances will have their own `Contact` instance for all the targets that have been detected by one or more sensors in the task group. All the instances of the `CIC`, the defending units, have one instance of the `ContactManager`, responsible for bookkeeping of all the contacts held by that unit.

When a sensor detects a target, the sensor is established as a listener to the target; and both the sensor and the `CIC` will have a reference to the target instance. Consequently it is possible to ask the target for its state variables like its course, speed or even destination. It is of utmost importance <u>not</u> to take advantage of this possibility in the simulation. For example, if we allowed the sensor or CIC to ask for the next waypoint of

the mover, we would jeopardize the realism of the model, by allowing our computer code to be far more capable than any known realistic sensor. In the `kitchen` package we are only allowed to ask the target for its position and whether it is moving or not. The position is used to update the target in the internal coordinate systems, the boolean variable `moving` indicates whether the target is "alive" or not. This rule is based on the fact that only very few sensors are capable of presenting any target data of higher order than position. All other information available in a tactical data system or a navigational system are produced by integrating the changes in position over time. We want to replicate this closely.

1. The `Contact` and `ContactManager` Classes

`Contact` has three functions. First, it is a placeholder for all the information the ships need to associate with each incoming threat; second, it has methods for answering the necessary geometrical questions that may be asked by the decision makers. The third function is that instances of the class are used for reference in the data link procedures. Most of the attributes connected to a target in a combat information system in a ship will also be available in `Contact`.

Some of the features of the individual instances of `Contact` may be considered global attributes; that is, they are equal in every instance of the class that carries the same target. Examples of global attributes are the contact number, which of the `CIC`s have the tracking and reporting responsibility for the target, and the `CIC` responsible for acting on the target if it poses a threat. When information about a new target is received through data link, not through a sensor, the second constructor of the class copies the global information into the new instance. The method `processLinkData(Contact)` in `ContactManager` is responsible for doing the same whenever the contact is passed in a link message later on. The local attributes of the contacts are, of course, only of use to the CIC that created the contact initially.

a) Geometry

The class holds three methods crucial to all subsequent evaluation and engagement of the target. The idea is that once the bearing and range to the target from the ship has been set twice, separated by some time, the class is enabled to calculate all the required data. A nice side effect of doing it this way, separated from the main coordinate system but relative to the ship, is that corrections for own movements are automatically taken into account. The geometry methods are:

- `getCPA()` Returns (1) the minimum distance between the target and the ship that will occur (or has occurred) with the present relative velocity, and (2) the time to (or since) target CPA.
- `getRangeTimes(specified range)` Returns the time to (since) the mover crosses the circle at the specified distance from the ship. A similar method is described in detail for the detection sequence.
- `getDistanceNow()` Returns the present distance to the mover.

The `ContactManager` is a class designed to maintain a list of all the Contact instances held by a CIC. It will provide target information such as the most threatening contact, the contact corresponding to a specific target. The class also has methods designed to be used by the `SAMBattery`. The CIC has an accessor method to make its manager available.

2. Target Prioritizing

One basic philosophy of all SSM attacks is that the missiles launched should arrive in the target area during as short a time span as possible. This is played in the `kitchen` package also. As an immediate consequence, the defending ships will not be able to counter all the threats simultaneously if the number of SSMs exceeds the number of anti-air-warfare systems. We must expect that in all anti-air-warfare organizations there exist rules or tactics used to prioritize the incoming threats for SAM allocation.

In the `kitchen` package the targets are sorted by the defenders based on their threat level. The threat level is a number indicating how dangerous the target is assessed to be, and is used as the only means for queuing the targets for engagements by SAM. A number of factors are to be considered when the "dangerousness" of an incoming threat is evaluated.

- A target at short range is more dangerous than one far away.
- A target with high speed is a greater threat than one moving slowly.
- The targets coming directly towards a ship will appear more threatening than those with a substantial distance to the closest point of approach.

A convenient and realistic procedure for capturing the factors listed above is implemented in the package. When a target is first entered in the tracking system its threat level is set at positive infinity, but after two time separated observations, when geometry methods are available, its value is calculated as follows:

- Threat level is the number of seconds until the missile will hit the target ship if the CPA distance is zero. In this case, the SSM is aiming directly for the ship.
- If the SSM will not have a CPA distance equal to zero, the threat level is the number of seconds to CPA plus two times the number of seconds it would take the missile to move from the CPA to the ship with its present speed.

If at target is shot down or fails to fulfill its mission, it will be given the threat level positive infinity again. But to distinguish this situation from the initial seconds, a boolean variable is set indicating the target is dead. In addition, targets that are shot down are given the state variable for moving or not set to `false`.

3. The Data Link Procedure

In everyday AAW and almost all other types of warfare, it is unfortunate but common to have a level of uncertainty or confusion. In AAW there are two situations that are particularly demanding to the crew manning the sensors and combat information

systems. The first task is to classify the objects detected: Are the blips on the radar screen made by friendly aircraft or enemy aircraft? Are they perhaps coming from a civilian airliner from Saudi Arabia? The second issue is to have all contacts reported exactly once. Once a target is detected by one ship, information about the detection will normally be released on a data link system within few seconds. If a second unit detects the same target and the discipline is less than perfect, chances are the second ship will also release the target on link. If there is a third unit listening to the same channel of information, this unit will wrongfully believe there are really two targets in the area if this latter unit does not cover the target area with own sensors.

As a means of addressing the challenges described above, the data link systems were developed for transferring information between units in near real time. Systems of data links today are a least common denominator for ships working together, and we are playing along the same lines in the `kitchen` package. The way information is gathered and disseminated in the package, however, outperforms reality to some extent. In this model there is no risk of misinterpreting a tracked contact. Through the scheme of neutral instances connecting the targets and the sensors, there is no risk of accidentally classifying a friendly unit as a target. However, such possibilities could relatively easily be built into the model, since most of the infrastructure is already in place. This model is not designed to evaluate the identification challenge presented to naval units, but to compare two different tactics for launching SSMs. The model is not developed into higher resolution in the area of target identification based on one of two assumptions. It is assumed that either the scenario of the incoming SSMs occurs when no friendly aircraft are in the area, or if there is the possibility of friendly aircraft the effects of the complicated air picture will have equal leverage under all scenarios. The data link rules in the `kitchen` package are summarized as follows:

- When a target is initially detected it is immediately released as a data link event.
- Reception of this event prevents later detection of the same target, but by other sensors, which would result in transmission of a new link event.

- The unit first reporting the target will maintain reporting responsibility for it as long as it is held by at least one sensor.
- If the target changes velocity the reporting unit informs the other units.

4. Removing Targets

Once detected, all targets will eventually disappear from the coverage area of the sensor. The disappearance could result from several effects: The range could get too large, the target could descend below the horizon, or it could fall to the ground or sea. All these effects are modeled in the `kitchen` package. The first unit that detects a new target automatically assumes responsibility for tracking the target and for reporting its whereabouts on link. Other units may detect the target later on, but after hooking up as listeners they will not report the target. If the tracking unit looses the target from its sensors, it will inform its peers and its superior. When the `OTC` is informed about a lost target it will place the corresponding contact on hold for removal. If another unit is tracking, or gains track of the lost unit, it will take over the responsibility. If this happens, the OTC will remove the target from hold for removal. If no unit has reported to be tracking the target by the end of the holding period, the `OTC` will order all its subordinates to discard the contact instance associated with the lost target. If the lost target is redetected, it will be wrapped into a new instance of the `Contact` class. There is a marginal possibility that a target that reappears in the detection range very close in time to its removal will be double tracked. This was initially regarded as an error in the code, and the correction was available. It is however quite realistic that such a situation will occur and the code has been left unchanged. The consequence may be, and exercises show this as a real possibility, that unnecessarily many weapons are launched at this particular target.

E. THE ENGAGEMENT PROCEDURES

After discussing moving entities and the detection process with associated information exchange, we will now discuss the third principal area of the model in this thesis - the engagement. This section will describe the rules that are built into the package for dealing with the threat of incoming missiles. As described earlier, there are two levels of deliberation in the air defense: First, when time allows is the coordinated use of force assets, like SAMs; and second is the individual ship using any weapon in the act of self-defense. In the `kitchen` package, distinction between the two is made by use of the threat level associated with each particular target. Targets subject to immediate attack are those with a current threat level below a userspecified threshold. All other targets are subject to coordinated attacks, ordered by the `OTC`.

Before we enter the air defense procedures however, we will address the launching of the SSMs, the weapons that are threatening the ships in the first place.

1. Attacking with Surface-to-Surface Missiles

All effective tactics for attacking with SSMs, and probably any weapon, suggest that concentration of forces in time make it harder for the enemy to counter the attack. This principle is built into the `kitchen` package when the attacking side is launching their SSMs.

The procedure starts with giving the `SOFPB` a time for the SSMs to hit their target. The senior officer passes this time on to her subordinates and they will individually calculate when they should fire their first SSM to achieve a hit in the target at the correct time. It is assumed that the individual missile launchers, `FPB`, know the position and velocity of their targets. Based on in this information and the route each individual missile shall traverse, with or without waypoints, the individual launch times are scheduled. Common to many missile-launching systems is a minimum time separation between two consecutive firings from the same battery. This limitation is

modeled; if two SSMs have equal distance to their target, and thereby should have identical launch times, a minimum time separation is enforced.

The missiles launched from the `FPB` are of the class `SSMissile` and their performance as flying objects is described in Section 4.b. Particular to the SSMs are two altitudes specified in the constructor as well as parameters for their search window. The first of the two altitudes is the transit altitude of the missile. It will climb to this height after launch and stay at this height until the target is detected. When the target is detected, it will adjust it altitude to what is called the terminal altitude and maintain this until it hits the target. This gives the user the ability to model SSMs that are seaskimming in their final phase toward the target.

The parameters for the target search are two times. The first is the number of seconds prior to predicted impact that the seeker will go active, and the second is the number of seconds the seeker is active. For example: A SSM has a predicted hit in second 1000, and it has seeker times 80 and 70 and heights 0.043 and 0.0045. The missile will climb to altitude 248 feet after launch (0.043 nautical miles = 248 feet = 80m) and continue at that altitude. At second 920 (80 seconds prior to hit) the seeker starts looking for the target. The target detection process itself is not modeled, but the time it takes the seeker to detect the target is drawn from a distribution specific to the missile we want to model. If the number drawn is greater than 70 (In this example the seeker active period is 70 seconds.) the SSM will not detect the target and it will ditch at the end of the seeker window. If the number drawn is 50, say, the missile will continue with its present velocity and altitude up to simulated time 970 when it detects the target. At time 970:

- The SSM will descend (or climb) to its terminal altitude.
- It will become a listener to its target.
- An instance of the `OutcomeObserver` will be constructed.
- The SSM will adjust its heading to intercept the target correctly, both initially and also later on if the target should change velocity.

When SSMs are fired at short range, situations may occur where the flight time is shorter that the active time for the seeker. The design does not allow the SSMs to activate their seeker before the missile has reached its cruising altitude and speed, a time interval called `settleTime`. If the time between predicted impacts and launch minus the time to settle (the available time) is less than the time the seeker is active, the user is given two options:

- Compare the available time directly with the random draw from the chosen distribution of time to acquire the target. This procedure implies declining performance of the SSM at short distances, and the probability of acquire will go down.

- Compare the available time with a reduced random time to acquire. The reduction factor is equal to the ratio of time available over seeker time. This procedure implies less or no decline in probability of acquiring the target.

If the SSM is not shot down, it will advance towards its target at its terminal altitude until its position is within a specified position accuracy of the target. At this point the missile will ask its `OutcomeObserver` for the result of the incident. Based on the probability of hit that is specified for the SSM and a Uniform (0.0, 1.0) random draw, the outcome observer will decide if the SSM was successful or not.

2. Selection of Firing Unit

At the moment the OTC decides to launch a deliberate engagement, it will go through a process of polling all of its subordinates to determine which one is the most suitable for executing the counter attack on the particular target of interest. The method used to quantify the availability of a particular CIC to engage a particular threat is in the CIC class, `getAvailability(Contact)`. The code is presented in Table 4.3.

```
public double getAvailability(Contact refContact){
   Contact c = manager.getContact( refContact.getContactNumber() );
   double availability = 0.0;
   if(c != null && targetsOnHold.size() <= battery.length){
      availability = getMissileLoadLeft();
   }
   if(availability > 0.0 ){
      availability += Math.max( 0.0, c.getRelativeSpeed()/c.getTargetRSpeed() );
      for(int j=0; j<battery.length; j++){
         if(battery[j].getNoMissilesLeft()>0){
            availability += Math.max(0.0,(1.0-c.getDistanceNow()/
                                                battery[j].getMaxRange()));
            if( battery[j].getIsEngagable(c) ){
               availability += 0.2;
            }
            else{
               if( battery[j].isInRangeGate(c) ){
                  availability += 0.1;
               }
            }
         }
      }
   }
   return availability/(1+(double)targetsOnHold.size());
}
```

Table 4.3. Method for Calculating a Units Availability

As with the threat level described in the previous section, the availability will be a number with no absolute meaning. This fact is allows us to relax normal procedures and hardcode two numbers into the method. The availability is only a measure of the suitability of choosing one unit, a ship, to counter a specific threat. The number produced by the method above will have significance only when compared to corresponding availability calculated for other ships. That is, the availability is a quantity with relative significance only.

The rules for calculating the availability are as follows:

- If a ship is requested for its availability of a target it does not hold the availability is zero.
- A unit is allowed to accept one more target than it has weapon systems. For example, if a ship has two batteries for launching SAMs, it may accept up to three targets. If the target is acceptable after this check, the availability is set equal to the portion of remaining missiles.
- Availability goes up proportionally to the positive ratio of relative speed, how fast is the target closing the ship, over how fast the target is moving in the local coordinate system. This results in assigning higher availability to directly

incoming targets than to targets moving otherwise, the unit the missile is aiming for will be the best suited to act on the threat.

- Availability is increased if the range to the target at present is less than the maximum range of the SAM-system. The more favorable ratio the larger the increase.

- Finally, availability is increased if the target is engageable directly or if it will be engageable at a later stage. The increment for directly engageable is twice the magnitude of the increase for future availability.

- At the end the availability is reduced proportional to how many targets is already on hold by this unit. Of two units with everything else equal the one with fewer targets waiting to be engaged will be the most available, hence also awarded the contract.

The availability-increase due to speed and range advantages are repeated for all the SAM batteries on the unit. Consequently, a ship with more systems will generally be preferred over one with fewer.

The reasoning for accepting exactly one more target for engagement than the count of SAM batteries is twofold. On the one hand, allowing more targets on hold would increase the complexity of the queuing process. An anti-shipping missile defense situation like the one we are modeling are highly dynamic, and it is paramount that the process of updating the data can be done with high accuracy and efficiency. One way of achieving this goal is to allow queuing to take place in only one place. In the package the number of detected and incoming targets will exceed the number of weapons available to counter the threat, and the targets that can not be handled immediately are sorted and given priority based on their threat level. Now, with every change in a target state variables there is a potential change in the associated threat level and consequently the order of the prioritized targets may have to be altered. The `ContactManager` class does all this bookkeeping. Allowing each `CIC` instance hold long queues of targets to engage would increase the number of complex operations to handle if the `OTC` should decide to remove one target from one ship and assign it for another unit.

On the other hand, why are the CIC then allowed to take on anything but exactly the number of targets equal to the number of SAM batteries? This contradicts the argument above, but allowing this extra target makes the CIC more effective. When one engagement cycle is over, the SAM battery now idle will immediately be assigned a new task, we will avoid waiting for the OTC to go through a decision process before the battery (server) again goes active.

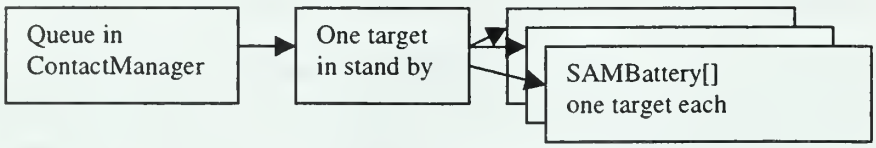

Figure 4.2. Queuing in the CIC

The advantages of reducing the turnaround time by having one target in reserve modus appear to compensate for the extra bookkeeping necessary. This is in fact a realistic setup since we would expect the OTC in a task group to provide the ships with one stand-by target if all the firing channels of the unit are full.

3. The Trackers

As with the sensors, the trackers that are modeled in this thesis are primarily using radar. The Tracker can be used to model a different kind of illuminator, but currently it is optimized to model radar trackers. Trackers are often referred to as illuminators, and in this context the phrases may be used interchangeably. The difference between a tracker and a radar used for detecting a possible contact is that the tracker uses all its energy in a very limited bearing and elevation at a time. While the beam of a normal search radar is 1 to 2 degrees wide and say 40 degrees high as it sweeps the horizon the beam from a tracker is known as a pencil beam and typically with less than two degrees width in all directions. Because of this design, the tracker is inefficient as a searcher. But when it is continuously looking at a particular target, it can provide high accuracy target data, and it will be able to observe target change of motion almost instantly. Instances of the

`Tracker` class are used for all the gunnery systems and the semi-active SAMs in the `kitchen` package.

When the tracker instances are constructed, they are given a reference to the ship on which they are placed as well as their open arc and center bearing. Open arc is the tracker's field of view; the relative sector where no superstructure or antennas are blocking the beam. Center bearing is the relative bearing in the middle of the open arc. The tracker is also constructed with a specific antenna height, to check whether its targets are above the horizon or not. Targets below the horizon cannot be tracked. Upon being ordered to track a target the following sequence is followed:

- The tracker checks if the target is detectable; i.e., within maximum range and above the horizon.
- If a target is detectable the tracker will draw a random number from the distribution describing the time delay it needs to lock on. Lock on is scheduled after this delay.
- At lock on, the tracker informs its superior that it is illuminating the target by a `TrackerEvent`.
- At lock on, the tracker hooks up as a listener to the target.
- Upon receiving information about changes of motion by the target, the tracker will investigate if the target is still possible to illuminate.
- If the target is no longer visible to the tracker, it will stop tracking and notify its customers by another `TrackerEvent`.

Only one target may be tracked at a time, and from the moment a tracker is ordered to follow a specific target it is inaccessible until it looses its target or is ordered to stop tracking.

4. Launching of Surface to Air Missiles

The SAM represents the cornerstone in the air defense of modern naval warfare, so the procedures for launching SAMs are carefully modeled in the package. The

`kitchen` package features two procedures for launching SAMs, done in the `SAMBattery` and `SAMBatteryPassive` classes, respectively.

The `SAMBattery` class involves launching of semiactive SAMs, such as SeaSparrow or Standard. Vital to systems modeled by this class is continuous updating of the target position and velocity by the firing unit. This is achieved by using trackers (illuminators) that need to "see" the target throughout the launch procedure and the flight time of the missile. According to this setup, the `SAMBattery` will not have completed its service until a) the last missile shot has physically passed the target or b) the target is shot down.

The `SAMBatteryPassive` class offers a simpler setup. Once the last missile in a salvo has left the ramp, the engagement is over and the battery is available for another assignment. Examples of systems falling in this category include Revolving Airframe Missile (RAM) or Mistral. This group of systems is commonly referred to as "fire-and-forget" weapons.

Even though the two separate classes simulating launch of SAMs are significantly different, all the weapons they launch are instances of the same class, the `SAMissile`. The neutral entity deciding whether an engagement is successful or not is also common, there will be one instance of the `OutcomeObserver` per pair of SAM and target.

a) Semi-Active SAM

Once the combat information center has been given an order to engage a target, or has decided to do so based on a self-defense situation, it will investigate its SAM batteries for availability. If there are more than one battery in the ship, the first one found will be chosen and awarded the contract even if subsequent systems may theoretically be a better choice. This simplification is left in the model because most ships are equipped with no more than one system of active or semi active SAM. And an overwhelming part those equipped with two or more systems have systems of identical performance but with different arcs of engagement.

If a SAM battery is capable of engaging a particular target, it may be given the order to do so by the method `engageTarget(Contact)`. The order will be given by direct invocation like any other order in the `kitchen` package; that is, the superior entity will invoke a method in its subordinate, not doing it by the property dispatching framework available in Modkit. The first step in the process will be to hand the target over to a tracker and wait for a positive response from this instrument. Feedback from a tracker will come as a `TrackerEvent` (see the section discussing event handling in the previous chapter), and the `SAMBattery` will have a method handling such messages. After being informed of a working tracker, the battery will set up the ordered salvo. Included in this procedure is collecting the number of SAM ordered to be launched and scheduling the individual launches. The delay from when the tracker reports it is ready to when the first SAM is actually launched is a function of two variables which may be fixed or random, as specified by the user. The first part of the delay should match the time it takes a missile in the launcher to get ready, time to train and elevate launcher if necessary and so on. The second delay should cover the necessary and realistic minimum time separation between two consecutive firings.

After the necessary delays, the method `doLaunchSAM()` is invoked. This method constructs an instance of the `SAMissile` class and starts it heading for the specified target. All the instances of the `SAMBattery` class require a `SAMissile` when it is constructed, called a prototype. When the `SAMBattery` launches a SAM, all the prototype's data are copied over to the new missile created. Critical in this process is establishment of the SAM as a listener to the target. Allowing the SAM to listen to its target resembles the real world situation of the SAM having continuous knowledge about the movements of the incoming target through one or more sensors. In the software package, the listening pattern models the tracking sensor. Because the active SAMs requires continuous support from a shipboard tracker, the `SAMBattery` is also listening to the SAM, as is the CIC of the firing unit. When the SAM is launched, it will construct an instance of the `OutcomeObserver` class. And when arriving at the position of the

target, it will request the neutral instance to decide if it was a hit or not. The result of the encounter is passed to its listeners.

The listener patterns established at the various stages in the process of launching SAM are illustrated in Figure 4.3, where the arrows point in the direction of the information flow. The base of the arrows are at the event source, where we will find the `generateXxEvent()` methods, the arrowhead points to the event listener, where we will find the `handleXxEvent()` methods.

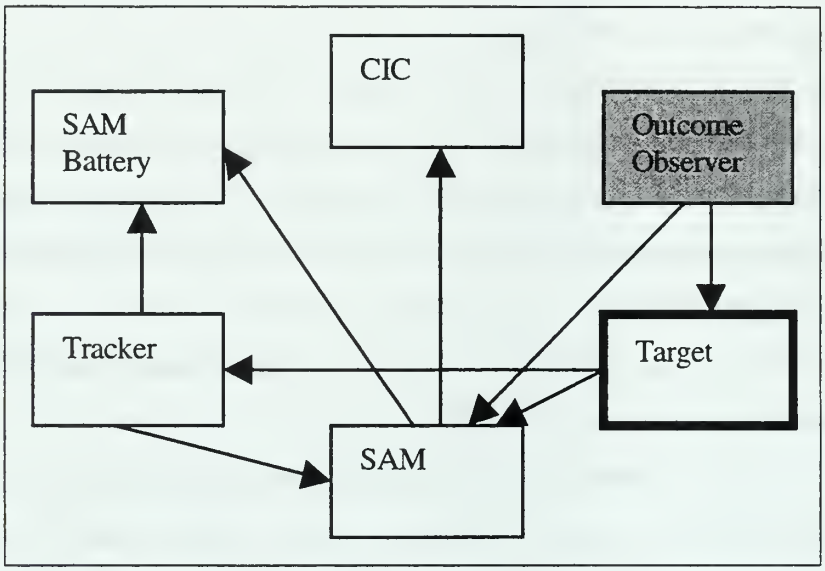

Figure 4.3. Listener Pattern in a SAM Engagement

The listener pattern described gives a realistic and efficient channel of information from the target to the tracker and to the SAM. It is the responsibility of the target, which is of the `SSMissile` class, to inform its listeners when it changes its velocity or its altitude. If the state change involves a new velocity, the event is of immediate interest to the SAM. It will have to adjust its own velocity accordingly to make sure it will hit its target. If the event is about a change in altitude, the tracker will check if it is still possible to "see" the target over the horizon. If the target has reduced its altitude and is now invisible to the illuminator, it will stop tracking and notify its listeners of this event. Upon

being notified that the tracker has stopped tracking its target, the SAM will no longer have guidance and it will be terminated, falling into the sea.

Once the SAM arrives at its target, it will request the result of the encounter from the neutral instance of the `OutcomeObserver` class. To make the decision, the observer will draw a Uniform(0.0, 1.0) random number and compare it to the hit probability of the SAM. The user specifies the hit probability for a SAM, and it is a function of the range, and the maneuvers done by the target during the time of flight:

- Hit probabilities for SAMs are specified by four parameters when the instance is constructed.
- The first parameter specifies the probability of a hit at the maximum range for the weapon.
- The second parameter gives a portion of maximum range where the probability of hit reaches its maximum. If this range is greater than zero, it is assumed that at any shorter range P(hit) stays at its highest value. Between the range specified by this parameter and P(hit) at full range, a linear reduction is applied.
- The third parameter gives the highest probability of hit.
- The last parameter is a factor to apply on the previously specified probability of hit if the SAM has to change its heading due to target maneuvers. The factor is applied in full if the necessary turn equals or exceeds 90 degrees, but reduced linearly to zero for smaller turns.

Say a SAM is chasing a target at 90% of its maximum range and that the specified parameters are pHit[0] = 0.5, pHit[1] = 0.8, pHit[2] = 0.9, pHit[3] = 0.7. The probability of hit will be fixed at 0.9 for all targets out to 80% of maximum range and then fall to 0.5 at full range. At 90% of full range, the probability of hitting the target then is 0.7. Now, suppose the target changes its course sometime during the SAM flight causing the SAM to alter its heading 30 degrees. The reduction factor to apply with a 90 degree course change is pHit[3] = 0.7, but in this case we should only apply reduction factor 0.8 corresponding to 1/3 of the worst case reduction. After the necessary turn, the probability

of the SAM hitting its target has been reduced to 0.7 * 0.8 = 0.56 which is the number used by the `OutcomeObserver` instance to decide if the SAM was successful or not.

b) Passive SAM

In the `kitchen` package the `SAMBatteryPassive` class is responsible for launching passive surface-to-air missiles. Again the missiles launched are instances of the `SAMissile` class, inheriting from `SpaceMover` and `SurfaceMover`. The launching of passive SAMs is similar to launching semi-active missiles with two major differences.

The first difference between the two procedures reflects the different designs of passive and semi-active SAMs. The passive missile does not require an external sensor to illuminate the target, but it will utilize its internal passive sensor to detect energy transmitted from the target and aim for this source. In the package, the difference shows up in the method `engageTarget()` which is found in both `SAMBattery` and the `SAMBatteryPassive`. In the semi-active procedures, the next step is to order a tracker to lock on to the target. The process then halts until the tracker notifies that it is following. In the passive procedures, this step is omitted, but there is still a delay to model the time it takes the internal sensor in a passive SAM to achieve lock on to its target once it is pointed in appropriate direction and activated. From the moment the internal sensor locks on the target the launch procedure continues parallel to the description above up to the actual launch.

The second difference occurs at the moment the SAM is launched. The `SAMBatteryPassive` has no means of keeping in contact with the newly launched SAM and it is not hooking itself up as a listener. This is done in the `SAMBattery`, since this unit is always monitoring its missiles through the tracker.

For the semi-active SAMs we have the option to allow the missile to continue on to a new target after a miss provided the next target is in directly ahead and in

the beam of the current tracker. This option is not allowed with passive SAMs; if they should miss their target the missile is automatically lost.

5. Engaging with Guns

The `Gun` class models the gunnery systems in the `kitchen` package, and the neutral instance is the `GunMissileObserver` class. The `Gun` class is very generic, and it has been feasible to model both heavy equipment like a 5-inch gun and a lightweight 20-mm Close In Weapon System, CIWS, with the same class. Compared to the model of the SAM engagement procedures, the choice of design for gunnery systems has shifted much of the workload from the model of the weapon system over to the neutral instances. The decision to model it this way comes from the similarities between deciding whether a gun is hitting its target or not and deciding if a sensor detects an incoming object or not. There are many similarities between the algorithms in the `RadarMediator` and the `GunMissileObserver`.

a) The Gun Class

Like the SAM batteries and the trackers, the guns need a reference to the ship they are mounted on, as well as their open arcs and center bearing. Every gun may accept one target only, these entities have no means of holding subsequent targets in a queue while they are engaging. All guns require a lock-on from an associated tracker before opening fire. They are allowed to accept targets out to 120% of their maximum range. The assumptions here are that the time that elapses from an approaching target is at 120 and 100% of available range will be used to assign a tracker to the target and receive lock-on. Hence the gun should be ready to open fire at maximum range. The following procedures are implemented for engaging with a gun:

- The gun instance receives an order to engage by invocation of the method `engageTarget()`.
- The gun searches through its available trackers and orders the first available to start tracking.

- Upon receiving confirmation that the target is tracked, the firing solution is checked again. If the target is still outside range of the gun it waits; otherwise it opens fire after a specified delay. The delay should capture any manual operation necessary as well as training and elevating the barrel.

- The gun becomes a listener to the target when the tracker reports lock-on.

- When fire is opened, an instance of `GunMissileObserver` is constructed as a referee for this encounter.

- The gun will continue being occupied with this target until the engagement is cancelled by a superior entity, the tracker looses the target, the target stops (hit by someone else), the gun runs out of ammunition, or if it is informed by the observer that it has hit the target.

The gun will start off with some amount of ammunition available for the first engagement and it will also have associated with it a rate for refill and a rate for firing. The two rates will be employed when someone asks the gun how many rounds it has available at any given time. Refill starts simultaneously with engagement start and continues until the rounds available again reach the maximum level allowed.

The gun takes in two numbers to specify its single round probability of hitting the target. The first number is the probability of hit at maximum range and the last number is the probability of hit at zero range. Linear interpolation is done to find the actual hit-probability for intermediate ranges.

b) The `GunMissileObserver` Class

Two important assumptions were made when this class was designed:

- One hit is enough to declare the engagement a success. That is, after one hit the SSM will fail to reach its target.

- The individual rounds fired at the target are considered independent trials.

Following the assumptions, the engagement is viewed as a nonhomogenous Geometric process. That is, we want to find how many failures will

occur before the first success, the round that hits the target. It is nonhomogenous because the probability of success is a function of the range to the target. The following scheme is implemented:

- Based on target position and velocity and the guns maximum range, the `GunMissileObserver` calculates the amount of time the target is engageable using the same principles as in the sensor-mover geometry.
- Actual range to the target combined with the two parameters describing the hit probability from the gun determines the probability of hit.
- The minimum of (a) the rate of fire times the time available and (b) the number of rounds presently ready at the gun gives an upper bound on the number of rounds that will be fired in this engagement.
- The rate of fire combined with target relative motion (closing or opening) will determine how the probability of hit changes from one round to another.
- A threshold for the necessary probability of hit for this particular engagement is established by drawing a Uniform(0.0, 1.0) random number.
- Summation is done to find the number of misses we will see before 1- P(miss on all shots) equals or exceeds the threshold. The summation is however not done farther than determined by the number of rounds available.
- The number of rounds corresponding to exceeding the threshold value will be returned as the number of rounds to fire in the engagement, and the product of the number of rounds and the rate of fire will give the time for firing the successful round.
- At the time for firing the successful round, the time of flight for that round is calculated and the gun and its target is informed about the hit after this time of flight.
- The time of flight is found by assuming that the speed of the round is reduced to 50% of the muzzle velocity at maximum range, and using linear interpolation to find terminal velocity at present range. To find time to reach the target distances the average of muzzle velocity and velocity at target range is used.

As stated above, this procedure is very similar to the detection process presented in section C of this chapter. Although the detection process is continuous and

the process here is discrete, they turn out to be so similar due to the discretizing of the detection process that is done to accommodate the numerical integration in `RadarMediator`.

We have now worker our way through some of the most interesting pieces of the computer program in the `kitchen` package. There are more than 6000 lines of code in the package and the methods and procedures that could be briefed here represent only a sub-set of the total, but most of the mathematical procedures are discussed. For further investigations of the code the reader should visit the web-site maintained by the Operations Research department at the Naval Postgraduate School (http//web.nps.navy.mil/~ahbuss) which for a period of time will have a link to the `kitchen` package.

In the next chapters we will study one particular implementation of the package. The purpose of the next two chapters is twofold: first, the implementation is designed as an experiment to shed light on the question being asked in this thesis. And second, the trials will be a verification of the validity of the assumptions and procedures that have been made and explained in this chapter.

V. EXPERIMENTAL DESIGN AND CONDUCT

The primary measure of effectiveness (MOE) in this thesis is the number of successful SSMs under the different launch tactics. The launch positions were varied to achieve multiple directions of attack and attack distances, but the number of launched SSMs was constant throughout the scenarios. Some secondary experiments were run after the results from the first were evaluated. In these experiments, the most effective tactics found in the first were used in investigating separate, but related issues from the main experiment.

1. Split Plot Design

The experiment was conducted using a split plot design. The argument for using this design was mainly that the setup allows dividing the experimental error into two sources - either the error comes from the tactics chosen by the attacker or it comes from the capabilities of the defending forces.

The main effects were the firing directions of the SSMs and the launch distances. These effects are presumably under control of the Officer in Charge of launching the attack. The other source of errors is due to effects unknown to and beyond control of the attacking side in the encounter: What is the effectiveness of the defender's air defense, and what will be his choice of policy for launching missiles? Also, these two parameters are not available as unclassified data for calibration of the model by the author. The two sources of error will be utilized in the variation reduction scheme associated with split plot design (Hinkelmann and Kempthorne, 1994, Chapter 13).

2. The Primary Experiment

Three levels were chosen for each of the main effects - two extreme scenarios and an intermediate level for both treatments. The setup created nine whole plots. The nine combinations of three spreads of firing directions and three different firing distances are from here referred to as geometries. There will be four sub-plots representing the pairs of

one of two firing policies and an optimistic and a conservative estimate for the effectiveness of the weapons on the defending side.

a) The Geometries

For all the nine geometries, a total of 30 SSMs were shot at the same ships. The ships started out in the vicinity of position (4.0, 4.0) and headed due northeast. Their positions and formation at simulated time 1000 seconds are displayed in Figure 5.1. The firing distances used are based on the following pre-experimental observations.

- Long range is close to maximum range of the SSM we are approximating.
- Intermediate distance is within mean detection range for the sensors, but outside the range of any anti-air weapon.
- Short range is approximately at maximum range of the SAMs.

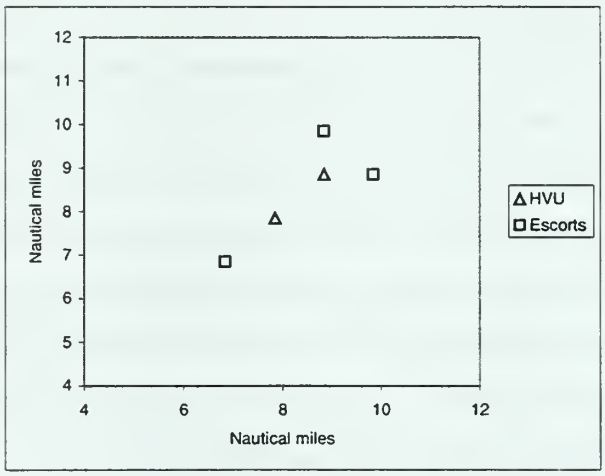

Figure 5.1. Position and Formation at Time of Missile Impact

The experiment was designed to have all missiles hit their target at simulated time 1000 seconds. In all the runs the 30 missiles were aimed at the same two high value units (HVU) with 15 missiles aimed at each. Figures 5.2 through 5.4 are three examples of the nine geometries that were used in the experiment.

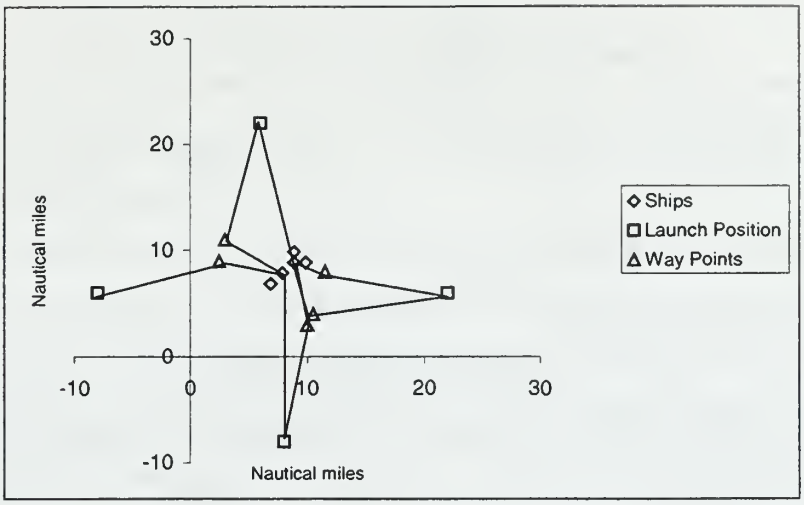

Figure 5.2. Launch Geometry with Intermediate Range and Medium Spread

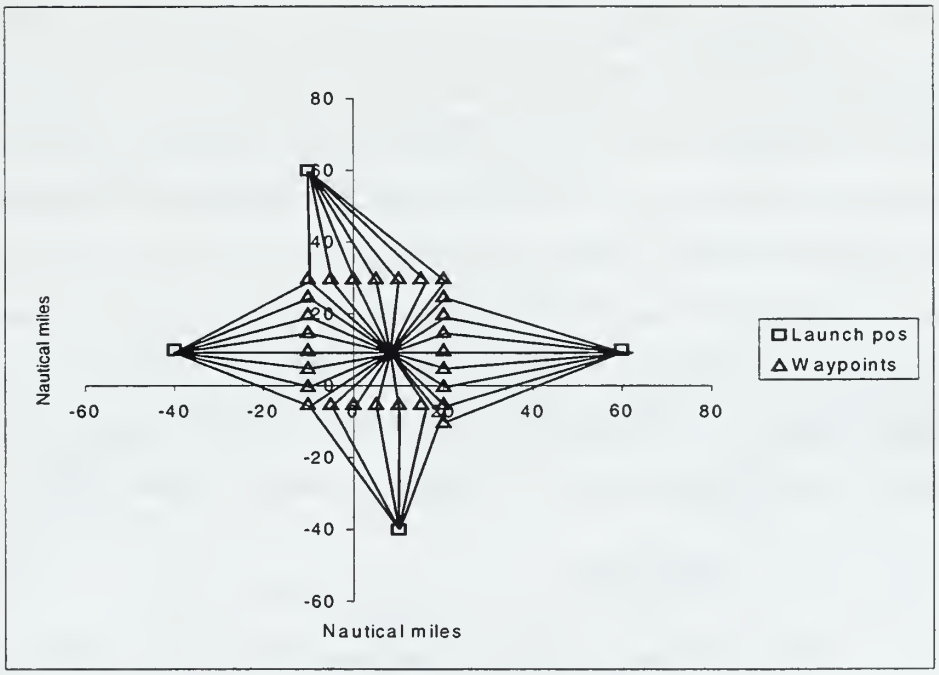

Figure 5.3. Launch Geometry with Maximum Spread and Long Range. The Target-ships are in the Center

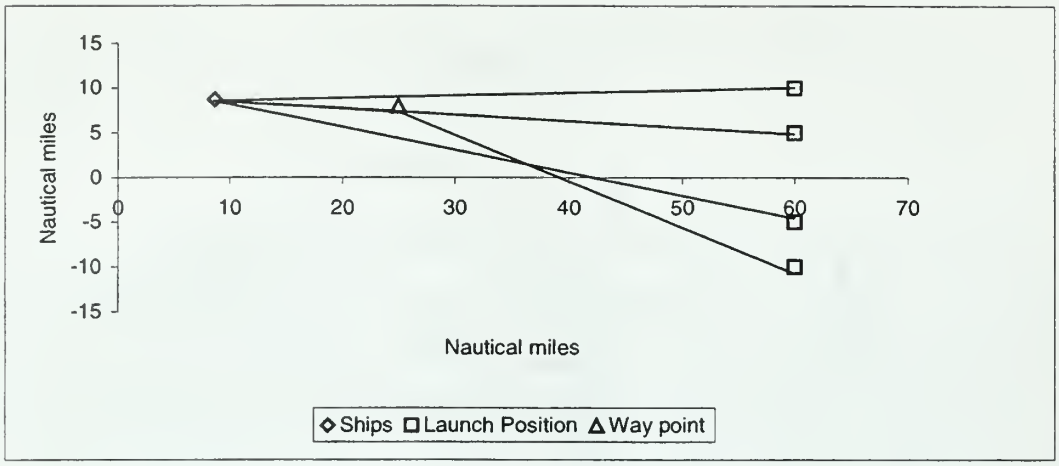

Figure 5.4. Launch Geometry with Minimum Spread and Long Range

b) The Firing Policies

The policies were shoot-shoot-look, SSL, or shoot-look, SL. Under the rules of policy SSL, every SAM launch will consist of salvoes of two missiles with a minimum time separation, and the second missile will be fired before the first is at the target. When the firing policy is SL, only one missile is launched at a time. The difference between the two should be apparent, firing two SAMs at every target will supposedly increase the probability of stopping that threat compared to firing only one, but on the other hand always launching two SAMs will initially reduce inventory twice as fast as launching only one. If the number of incoming SSMs is large, or subsequent attacks may be expected, policy SSL may result in too few or no missiles available later on.

c) Parameters

To find the parameters to use for the air defense weapons ships and weapons data were collected from open sources (Jane's Naval Weapon Systems, -), and an "interesting" mix of SSM launchers and targets were replicated. Some parameters are not available in open sources but nevertheless crucial to the outcome of the experiment. For example, we know that it takes some time for a radar to detect a target and it takes

even more time to establish the velocity at which the target is travelling. Also it is certain that when a situation as intense as a missile attack develops, decisions will have to be made and decision-making is never done in zero time. Other examples of parameters that are available only from classified sources are all the probabilities of success, like the probability of hitting the target with a weapon.

For the parameters that were inaccessible to the author, a series of educated guesses was made. For some of the data only a point estimate has been used, but for the most important parameters two values are guessed to attempt to bracket the true values. In the experimental setups, one of the two sub-plot effects is called performance. The performance reflects the parameters set by the author, and the two levels of the variable performance reflect what is assessed to be an upper and a lower bound for what the true values may be. Under the label "High" are all the data that would make the AAW highly effective, while under performance "Low" are parameters describing an inefficient missile defense. The following performance data have been used for ships and weapons. In **bold** are given parameters estimated by the author, data separated by a slant are for low / high performance of the air defense respectively.

(1) The Missiles

	SSM	Semi-active SAM	Passive SAM
Basis system	Harpoon	NATO Sea Sparrow (NSSMS)	Rolling Airframe Missile (RAM)
Initial speed (knots)	0.0	0.0	0.0
Cruising speed (knots)	550.0	1652.0	1321.0
Initial height (nautical miles)	0.0	0.0	0.0
Transit altitude	0.043 nautical miles, (80 meter)	Target altitude	Target altitude
Terminal altitude	0.0045 nautical miles, (8.3 meter)	Target altitude	Target altitude
Turn rate (radians/second)	**0.8**	**1.5**	**2.0**
Acceleration rate (knots/second)	**250**	**750**	**800**
Climb rate (knots)	**80**	**200**	**200**
Maximum range (nautical miles)	70	8.0	5.2

Probability of hit	**0.9** Given target is acquired	**0.4 / 0.6** At maximum range	**0.4 / 0.6** At maximum range
Maximum probability of hit	NA	**0.5 / 0.8**	**0.5 / 0.8**
Maximum P(hit) closer than (% of max range)	NA	**80**	**80**
Reduction factor of P(hit) with 90 degree course change	NA	**0.8 / 0.9**	**0.8 / 0.9**
Seeker activated prior to impact (seconds)	**80**	NA	From before launch
Seeker active (seconds)	**70**	Entire flight	Entire flight
Time for seeker to acquire	**Exponential, mean = 30 seconds**	NA	(Before launch) **Uniform, (3.6, 8.3) / (0.5, 2.0)**
May prosecute a new target after a miss	No	**Yes**	No

Table 5.1. Parameters for the missiles

The parameters in Table 5.1 are close to what open sources (Jane's Naval Weapons Systems, -) give as data for the Harpoon, the Sea Sparrow and the Rolling Airframe missiles.

The probability of an SSM acquiring the target within the 70 seconds available is 0.903 following from the exponential cumulative distribution function with mean 30 seconds. The conditional probability of hit given that the target is acquired yields a probability of a successful missile of 0.813. That is, if the SSMs are not shot down by the defenders they will hit their target with probability 0.813. The NATO Sea Sparrow missile (NSSMS) is a semi-active system that requires continuous tracking of the target by an illuminator onboard the firing ship. The Rolling Airframe missile (RAM) is a fire and forget system. Once the target is tracked by the internal sensors in the missile, it needs no further support from the launching ship.

We will be using the option of maintaining approximately the same probability of acquiring the target even if the time to settle for the SSM is shorter than the time the seeker is active. (See description of `SSMissile`.)

(2) The Defending Ships

Name	Max speed	Max turnrate	Accel-eration rate	Radar	Antenna height, feet	Tracker	SAM	Gun
Spruance	33	**0.03**	**0.5**	SPS 40b	94.0	SPG 60 Mk 95	NSSMS RAM	5 inch Mk 45 (2)
Meko	31	**0.03**	**0.5**	LW 08	66.8	STIR (2)	NSSMS	5 in Mk 45 Phalanx (2)
Doorman	30	**0.03**	**0.5**	LW 08	49.4	STIR	NSSMS	3 in Mk 100 Goalkeeper
Tarawa (2)	24	**0.02**	**0.6**	SPS 40b	140.0		RAM (2)	Phalanx (2)

Table 5.2. Ships and their Combat Systems

(3) The Radar Systems

Name (here)	Max range	Detection method		Detection factor		Full name
		High performance	Low performance	High performance	Low performance	
SPS 40	175	Inverse Quadratic	Inverse Quadratic	**3.6**	**0.6**	Lockheed SPS 40 e
LW 08	140	Inverse Quadratic	Inverse Quadratic	**3.6**	**0.6**	Signaal LW08

Table 5.3. Radar Data

(4) SAM Batteries

Missile	Delay to first launch		Separation in salvo		Ship	Missiles available	Open arc	Center bearing
	High perform	Low perform	High perform	Low perform				
NSSMS	**Uniform (4.0, 7.0)**	**Uniform (7.0, 10)**	**Fixed 1.5**	**Fixed 3.0**	Spruance	8	270	180
					Meko	16	360	na
					Doorman	16	360	na
RAM	**Uniform (0.5, 2.0)**	**Uniform (3.6,8.3)**	**Fixed 0.5**	**Fixed 1.0**	Spruance	21	300	180
					Tarawa (forward)	21	280	330
					Tarawa (aft)	21	270	150

Table 5.4. Parameters for SAM Batteries

(5) Tracking Systems for Guns and SAMs.

Name	Max range	Delay to detection, seconds		Ship	Associated weapon	Antenna height (feet)	Open arc	Center bearing
		High performance	Low performance					
SPG 60	80.0	**Uniform (3.0, 5.0)**	**Uniform (5.0, 9.0)**	Spruance	5 in Mk 45	103	310	000
Mk 95	20.0	**Normal μ=4.0 σ=0.6**	**Normal μ=6.0 σ=0.8**	Spruance	NSSMS	64	300	180
STIR	76.0	**Uniform (3.2, 5.7)**	**Uniform (5.0, 7.7)**	Meko (forward)	NSSMS 5 in Mk 45	62.8	330	000
				Meko (aft)	NSSMS 5 in Mk 45	55	320	180
				Doorman (forward)	NSSMS 3 in Mk100	61.8	320	000
				Doorman (aft)	NSSMS 3 in Mk100	61.0	320	180
Phalanx	3.0	**Uniform (0.2, 0.6)**	**Uniform (1.0, 3.0)**	Meko (forward)	Phalanx	40.5	270	000
				Meko (aft)	Phalanx	48.5	270	180
				Tarawa (forward)	Phalanx	98.5	270	040
				Tarawa (aft)	Phalanx	60.5	270	220
Goal-keeper	7.0	**Uniform (0.2, 0.7)**	**Uniform (1.2, 3.8)**	Doorman	Goalkeeper	40.5	290	180

Table 5.5. Parameters for Tracking Devices

(6) Gun Systems

System	Max range	Initial velocity	Rounds pr sec	Refill pr second	Rounds ready	P (Hit)		Delay to fire		Ship	Open arc	Center bearing
						High performance	Low performance	High performance	Low performance			
5 in Mk45	8.2	1570	0.3	**0.05**	20	**0.08** **0.23**	**0.05** **0.13**	**Uniform (4.0, 11.0)**	**Uniform (8.0, 15.0)**	Spruance (forward)	300	000
										Spruance (aft)	300	180
										Meko	300	000
3 in Mk100	6.5	1798	2.0	**0.1**	70	**0.07** **0.15**	**0.04** **0.10**	**Uniform (3.0, 9.0)**	**Uniform (6.0, 12.0)**	Doorman	280	000
Goal-keeper	1.1	2002	50	**20.0**	1190	**0.03** **0.08**	**.009** **.016**	**Uniform (0.2, 1.0)**	**Uniform (0.3, 1.5)**	Doorman	280	180
Pha-lanx	.81	2002	70	**20**	1470	**0.02** **0.07**	**.010** **.020**	**Uniform (0.2, 0.8)**	**Uniform (4.0, 11.0)**	Meko (forward)	260	000
										Meko (aft)	310	180
										Tarawa (forward)	270	040
										Tarawa (aft)	270	220

Table 5.6. Parameters for Gun Systems

(7) Decision and Classification Delays

Unit	Evaluation delay		Decision delay		Threatlevel self defense	Threatlevel deliberate
	High perform	Low perform	High perform	Low perform		
OTC	**Normal** μ=5.0 σ=1.0	**Normal** μ=5.0 σ=1.0	**Uniform (8.0, 15.0)**	**Uniform (8.0, 15.0)**	**40**	**350**
Spruance	**Normal** μ=5.0 σ=1.0	**Normal** μ=5.0 σ=1.0	**Uniform (8.0, 15.0)**	**Uniform (8.0, 15.0)**	**40**	**350**
Meko	**Normal** μ=5.0 σ=1.0	**Normal** μ=5.0 σ=1.0	**Uniform (8.0, 15.0)**	**Uniform (8.0, 15.0)**	**40**	**350**
Doorman	**Normal** μ=5.0 σ=1.0	**Normal** μ=5.0 σ=1.0	**Uniform (8.0, 15.0)**	**Uniform (8.0, 15.0)**	**40**	**350**
Tarawa	**Normal** μ=5.0 σ=1.0	**Normal** μ=5.0 σ=1.0	**Uniform (8.0, 15.0)**	**Uniform (8.0, 15.0)**	**40**	**350**

Table 5.7. Parameters for Decision and Classification Delays

(8) Detection Factors. Preliminary runs were done with the sensors and the targets as the only players to ensure reasonable values for the detection factor, α. In the following setup, two SSMs are fired at targets about 60 nautical miles away and each of them would navigate through one waypoint. The waypoints are at about 14 and 18 nautical miles from the ships, hence detection will most frequently take place prior to the SSM reaching this point. Prior to reaching the waypoints the SSMs are not heading directly for their targets. The largest detection range possible for the setup is 34.03 nautical miles, given by the antenna height for the sensor and the SSM altitude. The following data represent ranges for a total of 2400 detections.

Detection method	alpha	Data range	Mean	Variance
Inverse Quadratic	0.6	9.19 , 33.81	20.68	27.10
Inverse Squared	3.7	4.67 , 33.87	20.54	45.82
Inverse Quadratic	3.6	17.27 , 34.03	27.77	13.80
Inverse Squared	14.4	16.42 , 34.03	27.62	19.18

Table 5.8. Sample Detection Distances

Detection method "Inverse Squared" indicates that $n = 2$ in Equation 4.4, corresponding to a detection rate proportional to the inverse squared distance between the sensor and the target. Method "Inverse Quadratic" corresponds to $n = 4$, where the detection rate depends on the inverse fourth power. The values of α are chosen to give approximately equal means for the two methods. Table 5.3 states that the detection method that is used in the experiment is "Inverse Quadratic" throughout. This is consistent with spherical spreading of both the transmitted radar pulse and its echo. The data from the inverse squared method is presented here for illustrative purposes only.

In this chapter we have described the experimental setup and the calibration of the model. The choices of parameters for the individual systems are a result of available unclassified data and the author's estimation.

The choice of type and number of defending ships and number of SSMs to be fired is done to create some interesting results in the next chapter. We attempted to

have the attacker and the defender reasonably balanced, in the sense that neither almost all, nor almost none of the SSMs hit their target. To be able to present the upcoming results, over 20 hours of computation time were necessary.

VI. RESULTS

The setup as described in Section 5.2 required a total of 36 runs of the model; after each run the following data were collected :

- The number of the following possible outcomes for SSMs: Hit, Shot down by SAM, and Shot down by Gun. The difference between the number of hits and the number shot down is the result of the SSM missing the target.
- The number of SAMs launched, collected separately for the two types present, NSSMS and RAM.

The parameters were changed over the 36 runs according to the three treatment levels for each of spread and distance, creating nine whole plots. Within each whole plot two levels each of performance and policy creating four sub-plots for each whole plot. The following variables were kept constant throughout all the runs of the model:

- The group of defending ships; i.e., the number of participants and specifications of the individual units.
- Initial position, course, speed and formation of the defending ships.
- Resources available to the defending ships (sensors and weapons).
- Rules for interaction and information distribution within the defending organization.
- The number of SSMs launched in total and at the individual targets.

The first steps in the analysis are focused on the number of SSMs that hit their target - this number is our measure of performance. The first two sections of this chapter will evaluate the influence of the choice of spread of firing axis. The last section will discuss matters of interest to the defending ships after the facts about the various launch geometries are established.

A. SUMMARY STATISTICS AND GRAPHICAL DISPLAYS

The main experiment yielded 1620 lines of data, resulting from 45 replications of each of the 36 different combinations of treatments. Recall that for each run 30 SSMs were fired.

First of all the means for each of the subplots were computed.

		Maximum Spread		Medium spread		Minimum spread	
	Air defense	SL	SSL	SL	SSL	SL	SSL
Short Range	High performance	5.00	4.60	4.00	4.18	6.51	5.67
	Low performance	12.93	11.62	12.31	10.82	14.80	13.00
Intermediate range	High performance	5.62	3.80	4.80	4.13	7.33	6.78
	Low performance	11.49	10.47	12.49	10.16	14.53	13.20
Long range	High performance	4.16	3.49	5.20	4.62	7.56	5.84
	Low performance	10.91	9.82	13.60	12.04	15.00	12.73

Table 6.1. Mean SSM Hit over all Sub-Plots

A few observations can be made by studying only the means. It was argued in the previous chapter that two different levels of the performance of the air defense would be reflected in two sets of parameters. The results are clear: high performance air defense reduces the mean number of SSMs that hit their target under all other variations of the scenario. It should also be observed that the difference between firing policies SL and SSL are generally small, but in 35 of 36 sub-plots firing policy SSL makes it harder for the attacker. Under all but the medium-spread, short-range, high-performance geometry policy, SSL has a slightly smaller number of successful SSMs. The most important observation from this set of data, however, is the fact that under both levels of performance and of policy, the whole plot with minimum spread has the most favorable values for the number of SSM that hits their target. This is an important observation, directly addressing the main goal of this thesis. Among the two other whole plots, maximum spread appears most efficient at minimum range while medium spread is better

at long range. For now, however, it is too early to tell if any of the observed differences have practical significance. Figures 6.1 and 6.2 presents the same data as Table 6.1 simplified in Figure 6.1 to the nine whole plots and by combining subplots of firing policy SL and SSL. And in Figure 6.2, the 18 means are shown which result when the variable policy is ignored.

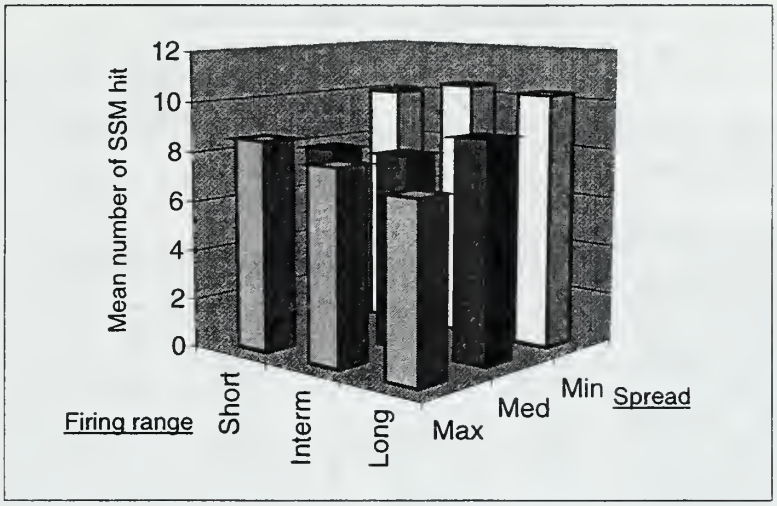

Figure 6.1. Mean Number of SSM Hits for the Whole Plots

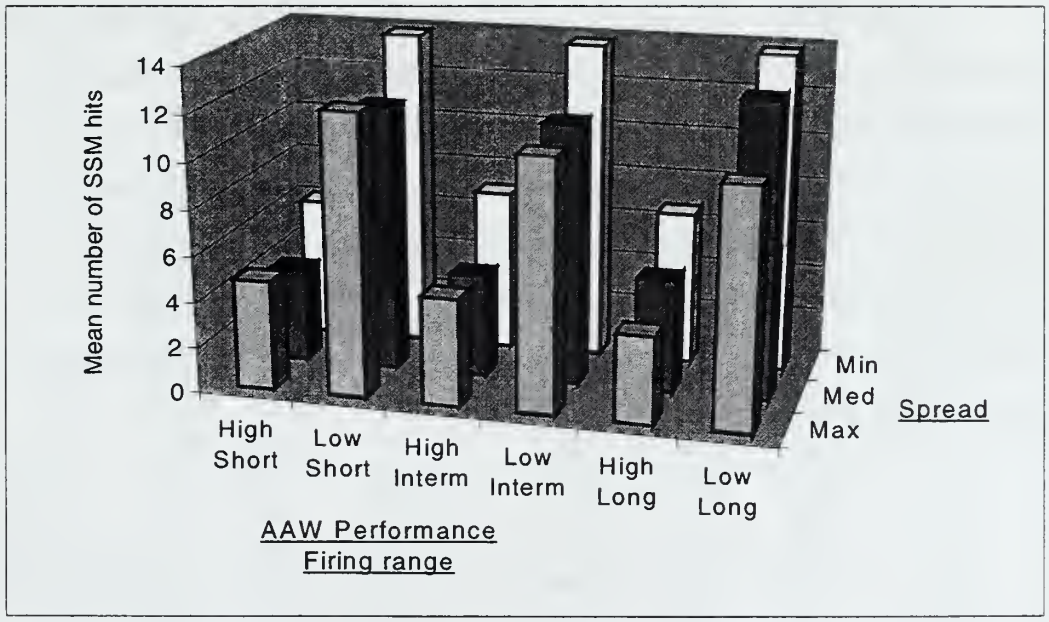

Figure 6.2 Mean SSM-Hit over Spread, Performance and Distance

Our next goal is to develop the standard deviations for the 36 sub-plots and to use this to construct confidence intervals, but first we will make a small divergence to investigate the precision of the data. While the absolute values of the standard deviations have only limited interest, the unitless coefficient of variation is more descriptive here. The coefficient of variation is the ratio of the sample standard deviation to the sample mean; values for our 36 combinations are given in Table 6.2.

		Maximum Spread		Medium spread		Minimum spread	
		SL	SSL	SL	SSL	SL	SSL
Short Range	High performance	0.377	0.438	0.449	0.425	0.418	0.365
	Low performance	0.211	0.166	0.265	0.204	0.158	0.194
Inter mediate Range	High performance	0.346	0.440	0.391	0.333	0.298	0.333
	Low performance	0.218	0.243	0.191	0.245	0.157	0.194
Long Range	High performance	0.435	0.542	0.340	0.433	0.327	0.292
	Low performance	0.247	0.238	0.183	0.203	0.193	0.196

Table 6.2. Coefficient of Variation for SSM Hit over all Sub-Plots

The coefficient of variation is an indicator for the spread of the data relative to the mean, and it is often used to dynamically decide the number of replications necessary in a simulation to achieve a desired level of precision (Law & Kelton, 1991, pp 536-540). In our case, the experiment was conducted with fixed sample size for all sub-plots to achieve balanced data for subsequent analysis of variance. We may however use the same procedure to establish a lower bound for the precision γ in our experiment. Here $t_{n,p}$ is the critical value from the t distribution with n degrees of freedom, s is the sample standard deviation and $\overline{X}$ is the sample mean.

$$\gamma \leq t_{n-1,\alpha/2} \frac{s}{\overline{X}} \frac{1}{\sqrt{n}} \qquad (6.1)$$

In our case $n = 45$, the corresponding t-value is 2.015 and our highest coefficient of variance is 0.542. At our least stable subplot, the achieved relative precision is 0.162. To have "small relative error" (Law & Kelton, 1991, p 540) it is suggested to increase the

number of replications such that $\gamma \leq 0.15$. In our case this would imply "small relative error" in all subplots with coefficient of variation less than 0.499 which is also the case in all but the one worst case just discussed.

The next table is an intermediate step to clarify the picture of the confidence intervals. In the table the 36 sub-plots are ranked according to the mean number of successful SSMs. The confidence intervals are pointwise and based on the assumption that 45 replications are sufficient to allow use of the Normal distribution; i.e., the Central Limit Theorem is applicable. The confidence intervals are at the 95% significance level and two sided.

Index	Distn	Spread	Perform	Policy	Mean hit	Pointwise		Simultaneous	
						Upper 95% CI	Lower 95% CI	Upper 95% CI	Lower 95% CI
1	Long	min	Low	SL	15.000	20.668	9.332	24.225	5.775
2	Short	min	Low	SL	14.800	19.370	10.230	22.237	7.363
3	Interm	min	Low	SL	14.533	19.007	10.060	21.814	7.253
4	Long	med	Low	SL	13.600	18.480	8.720	21.542	5.658
5	Interm	min	Low	SSL	13.200	18.208	8.192	21.349	5.051
6	Short	min	Low	SSL	13.000	17.944	8.056	21.047	4.953
7	Short	max	Low	SL	12.933	18.291	7.575	21.653	4.214
8	Long	min	Low	SSL	12.733	17.614	7.853	20.676	4.791
9	Interm	med	Low	SL	12.489	17.154	7.824	20.081	4.897
10	Short	med	Low	SL	12.311	16.639	7.983	19.355	5.267
11	Long	med	Low	SSL	12.044	16.845	7.244	19.856	4.232
12	Short	max	Low	SSL	11.622	15.413	7.832	17.791	5.453
13	Interm	max	Low	SL	11.489	16.391	6.587	19.467	3.511
14	Long	max	Low	SL	10.911	16.194	5.628	19.509	2.314
15	Short	med	Low	SSL	10.822	17.212	4.432	21.221	0.423
16	Interm	max	Low	SSL	10.467	15.457	5.477	18.588	2.346
17	Interm	med	Low	SSL	10.156	15.028	5.283	18.085	2.226
18	Long	max	Low	SSL	9.822	14.405	5.239	17.281	2.364
19	Long	mini	High	SL	7.556	12.403	2.708	15.444	0.000
20	Interm	mini	High	SL	7.333	11.615	3.051	14.302	0.365
21	Interm	mini	High	SSL	6.778	11.198	2.358	13.971	0.000
22	Short	mini	High	SL	6.511	11.840	1.182	15.184	0.000
23	Long	mini	High	SSL	5.844	9.186	2.503	11.283	0.406
24	Short	mini	High	SSL	5.667	9.718	1.615	12.260	0.000
25	Interm	max	High	SL	5.622	9.436	1.809	11.829	0.000
26	Long	med	High	SL	5.200	8.661	1.739	10.833	0.000
27	Short	max	High	SL	5.000	8.691	1.309	11.006	0.000
28	Interm	med	High	SL	4.800	8.481	1.119	10.791	0.000
29	Long	med	High	SSL	4.622	8.549	0.696	11.012	0.000

30	Short	max	High	SSL	4.600	8.551	0.649	11.030	0.000
31	Short	med	High	SSL	4.178	7.656	0.699	9.839	0.000
32	Long	max	High	SL	4.156	7.700	0.611	9.924	0.000
33	Interm	med	High	SSL	4.133	6.829	1.438	8.520	0.000
34	Short	med	High	SL	4.000	7.521	0.479	9.730	0.000
35	Interm	max	High	SSL	3.800	7.080	0.520	9.138	0.000
36	Long	max	High	SSL	3.489	7.194	-0.216	9.519	0.000

Table 6.3. Sub-Plots Ranked on Mean SSM Hits. Upper and Lower Bounds for Pointwise and Simultaneous 95% Confidence Intervals

Table 6.3 shows again what we have already deduced; the probability of a given SSM being successful is highest with a low-performance air-defense. The table also indicates that applying the Normal distribution may not be an appropriate model, at least not in the tail where the calculated lower confidence level is negative, an impossible value for the number of hits.

The index column of Table 6.3 corresponds to the x-axis in Figure 6.3. The middle curve in Figure 6.3 the represents the means, the curve immediately above and below the center are the pointwise and the two outer curves the simultaneous confidence intervals. Both sets of confidence intervals are at the 95% level. For some observations this simplistic model produces a negative lower confidence limit, here those values are the set to zero. The suggested negative values indicates that the use of the symmetric confidence intervals following the assumption of Normality is inappropriate, but the simplification will serve its purpose here at this initial stage where we just want to make some rough observations. The simultaneous interval is derived through basic probability theory and the assumption that the 36 individual observations are independent. The new individual alpha level to use is 0.00142, corresponding to the value that gives a combined probability of having all the 36 confidence regions cover their respective true means with probability 0.95 (Bonferroni intervals).

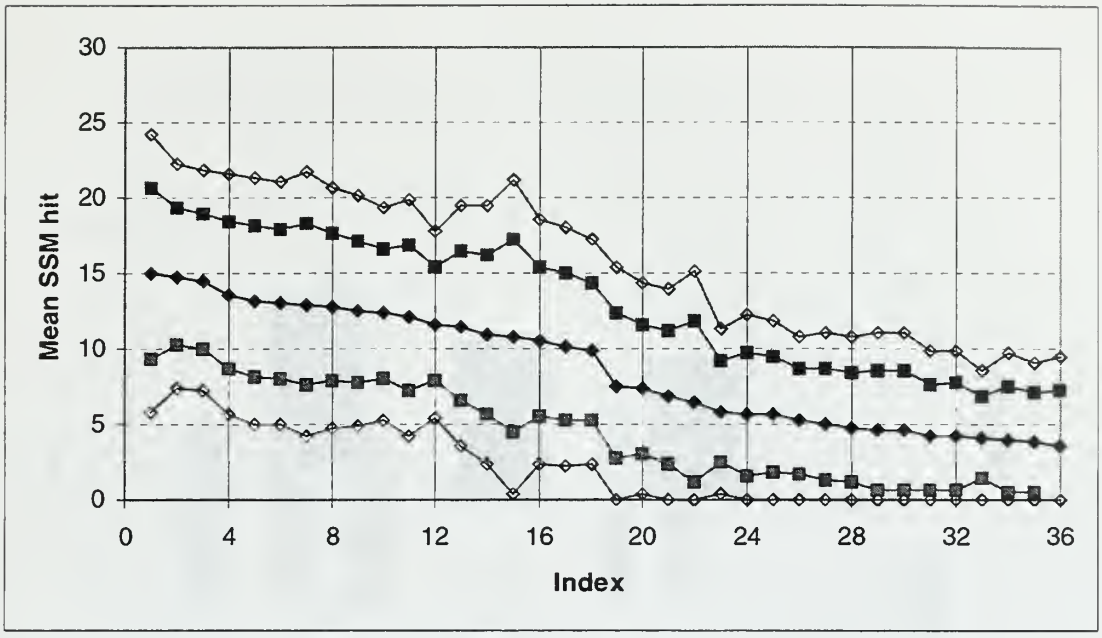

Figure 6.3. Mean, Pointwise and Simultaneous Confidence Intervals for Sub-Plots

Next we will create boxplots to investigate the variability and location of the number of SSM-hits for all data points, but separated for "maximum", "medium", and "minimum" respectively. The boxplots are made by use of SPLUS statistical package from Mathsoft Inc. using these settings:

- The data that falls within the box are from the second and the third quartile.
- Data between whiskers falls within 1.5 interquartile range of the median.
- The notch represents 95% confidence limits for the median.
- Data falling outside the whiskers are drawn separately.

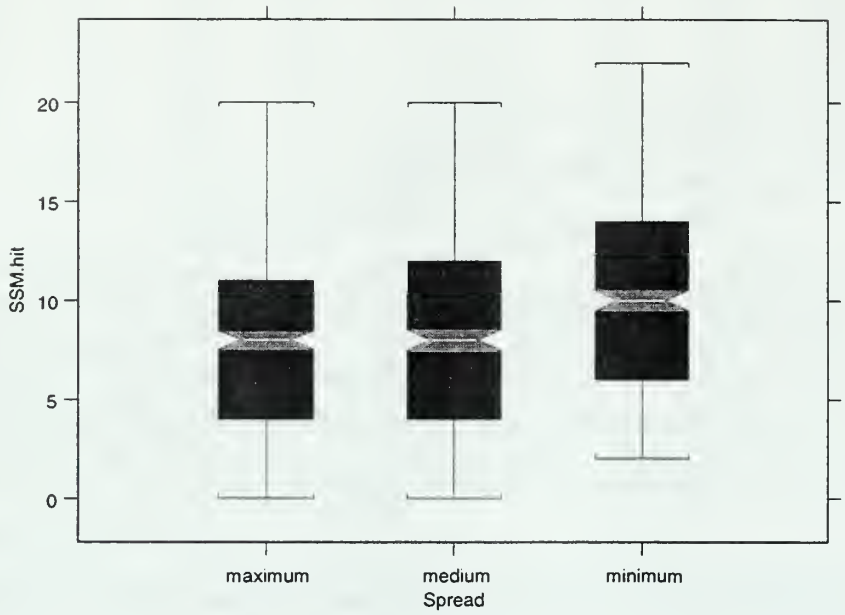

Figure 6.4. Boxplot of SSM Hits over Spread

The notches that overlap are an indication that the true location of the center of the distributions may be equal. In our case it appears the notches for "maximum" and "medium" overlap while "minimum" has a different center than the other two. Figure 6.5 shows boxplots over all geometries for levels of the three remaining factors.

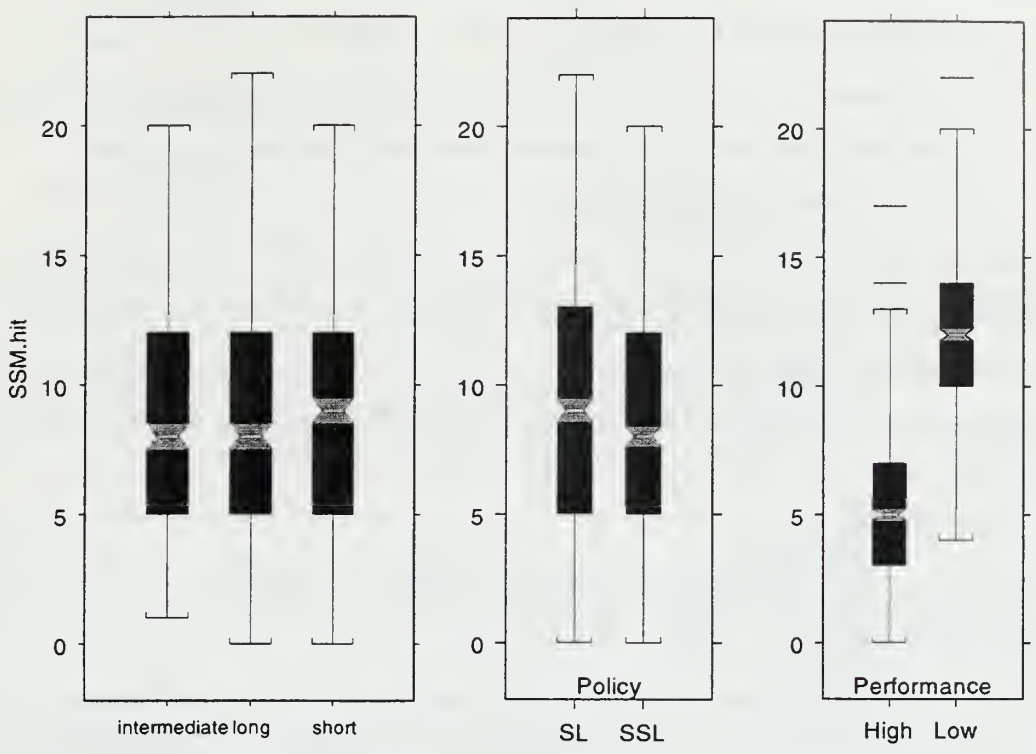

Figure 6.5. Boxplot for SSM Hit over Distance, Policy and Performance

Figure 6.5 further confirms the observations we inferred by studying the data alone.

B. ANALYSIS OF VARIANCE

We will now use analysis of variance (ANOVA) to further investigate the findings in the previous section. In the models we will take into account that there are two sources of error; one associated with the actions taken by the attacker and one associated with the defender. The errors that go with the defenders are experimental errors belonging to the firing policy and the performance of the defender air-defense. They are used to compute F-statistcs associated with the corresponding variables, Policy and Performance. The general error term is used to compute F-statistics for the two remaining whole plot

effects, the Spread and the Distance. The following summary statistics and ANOVA tables are produced by SPLUS. To ease the reader through the upcoming tables it is commented that the SPLUS terminology of the ~ (tilda) should be read as *modeled by*, a plus sign is read as *and,* and the colon as *interaction of*. Further, the variable mainExp is the data set comprised of the 1620 observations from the main experiment.

1. The Main Experiment

```
summary(aov(SSM.hit ~ Spread + Distance + Performance + Policy +
Error(Policy:Performance),data = mainExp))

Error: Policy:Performance
            Df Sum of Sq  Mean Sq  F Value     Pr(F)
Performance  1  20686.74 20686.74 325.2348 0.0352644
Policy       1    565.34   565.34   8.8882 0.2060294
Residuals    1     63.61    63.61

Error: Within
            Df Sum of Sq  Mean Sq  F Value     Pr(F)
Spread       2  1835.616 917.8080 170.2822 0.0000000
Distance     2     0.831   0.4154   0.0771 0.9258232
Residuals 1612  8688.556   5.3899
```

Table 6.4. Function Call and Output from ANOVA with All Observations Present

In our first ANOVA we model the number of SSM hits over all the subplots, the most general model available. Observe that under this model Performance and Spread are statistically significant effects for modeling the number of hits for SSM. Neither the Policy nor the Distance appears to have any significant influence on the expected number of hits. It is important to note that the model shows that the levels of the treatment "Spread" has relevance to the number of successful SSM, in the previous section we observed that minimum spread was advantageous to the attacker, and it is now possible to draw our first partial conclusions.

- Choice of spread in the firing geometry has significant influence on the expected number of hits in a salvo of SSMs. A minimum spread will give the largest number of hits.

- Distance has so far not showed any influence in any tool we have used to analyze the data. This is counterintuitive, and the issue will be revisited in the next chapter.

2. Analysis on Subsets of the Data

Having established that using minimum spread of the firing axis is advantageous to the SSM shooter, we will now examine the same question using subsets of the data: Is the advantage of concentration of firing axis significant under both level of policy and both levels of performance? After observing previously that the distance appears to have no influence we will not do any further investigations over the three levels of distance.

```
summary(aov(SSM.hit ~ Spread + Distance + Performance,
            data = mainExp[c(mainExp[,"Policy"]=="SL"),]))

             Df Sum of Sq  Mean Sq  F Value     Pr(F)
     Spread   2   1067.68   533.84   90.086 0.0000000
   Distance   2      3.20     1.60    0.270 0.7632872
Performance   1  11522.25 11522.25 1944.404 0.0000000
  Residuals 804   4764.39     5.93
```

Table 6.5. Function Call and ANOVA Table for First Subset

In the ANOVA presented in Table 6.6 we are investigating the effects of Spread, Policy and Distance for all observations with Policy is SL. The partition of the data is done in the `data = mainExp[c(mainExp[,"Policy"]=="SL"),]` statement. Because there is only one level of Policy present in the model we may drop the special error term from our main model.

The observations done under this model do not deviate from the ones done in the basic model - distance has no significance while Performance and Spread are significant. In Table 6.6 similar procedures are done investigating each of the levels of the sub-plot treatments separately.

```
summary(aov(SSM.hit ~ Spread + Distance + Performance,
data = mainExp[c(mainExp[,"Policy"]=="SSL"),]))

             Df Sum of Sq  Mean Sq  F Value     Pr(F)
Spread        2   779.351  389.675   80.305 0.0000000
Distance      2     9.040    4.520    0.931 0.3944097
Performance   1  9228.094 9228.094 1901.750 0.0000000
Residuals   804  3901.348    4.852


summary(aov(SSM.hit ~ Spread + Distance + Policy,
data = mainExp[c(mainExp[,"Performance"]=="Low"),]))

            Df Sum of Sq  Mean Sq  F Value      Pr(F)
Spread       2  1036.141 518.0704 78.27708 0.00000000
Distance     2    37.541  18.7704  2.83608 0.05924202
Policy       1   504.100 504.1000 76.16625 0.00000000
Residuals  804  5321.207   6.6184


summary(aov(SSM.hit ~ Spread + Distance + Policy,
data = mainExp[c(mainExp[,"Performance"]=="High"),]))

            Df Sum of Sq  Mean Sq  F Value      Pr(F)
Spread       2   830.884 415.4420 101.9897 0.00000000
Distance     2    24.240  12.1198   2.9754 0.05159115
Policy       1   124.844 124.8444  30.6489 0.00000004
Residuals  804  3274.990   4.0734
```

Table 6.6. Function Call and ANOVA Tables for Three Subsets of Data

We may observe that although the F-Value for Distance has gone up from the ANOVAs where the Performance is fixed, it is still insignificant at the 5% level. Both Performance and Spread shows significance at all three sub-models.

Based on the summary statistics, the plots and finally the preceding ANOVA we may summarize our findings under our main model so far:

- Spread is a significant factor for modeling the expected number of SSM hits for any course of action or effectiveness of the defenders air defense
- Concentration of firing axis should be the goal for the attackers.
- Firing policy shows significance in the models with the Performance fixed at one of its two levels.

C. SECONDARY EXPERIMENTS

The `kitchen` package was designed to answer questions concerning the effectiveness of different of attack geometries for SSM shooters. The findings in the previous sections make it possible to draw well-founded conclusions to this question.

Most of the modeling effort was devoted to the defense against anti-shipping missiles rather than the offensive side. The careful modeling of the defending ships makes some observations regarding their situation ready available after running the model. Also, some of the results in the main experiment are only partially in line with our expectations, and we will run some separate trials to explain these issues.

1. Consumption of Surface-to-Air Missiles

The defender's concern is opposite to the attacker's. In evaluating the cost of the air-defense defense, did it strain the resources too hard, or will the defender manage to withstand another wave of attack given that he survived the first? A critical measure of effectiveness (MOE) to the defender is the number of SAMs that was used in the defense. In our experiment variations in the number of SAMs that was used depended on all four treatment effects, but particularly on the choice of firing policy. We have seen that the choice of policy may be a significant factor, provided that we fix the performance-factor to one of its two levels. The question is whether the extra spending of SAMs was worth while.

To start pursuing this investigation we will make a simplifying assumption. The main experiment was run with two different weapon systems operating under the same firing policy. That is, both the NSSMS and the RAM batteries either fired one or two missiles in each salvo. In total there were 40 NSSMS and 105 RAM missiles ready in the whole force when the SSM attack was launched. Our simplification was to weight the two systems equally, hence assigning larger value to a single NSSMS missile than to a single RAM. In the upcoming analysis we will observe the fraction used as the measure of performance. Firing 20 NSSMS is of equal importance to the OTC as firing 52 or 53 RAM.

First we show that the difference in the number of SAMs fired under the two different policies using boxplots in Figure 6.6.

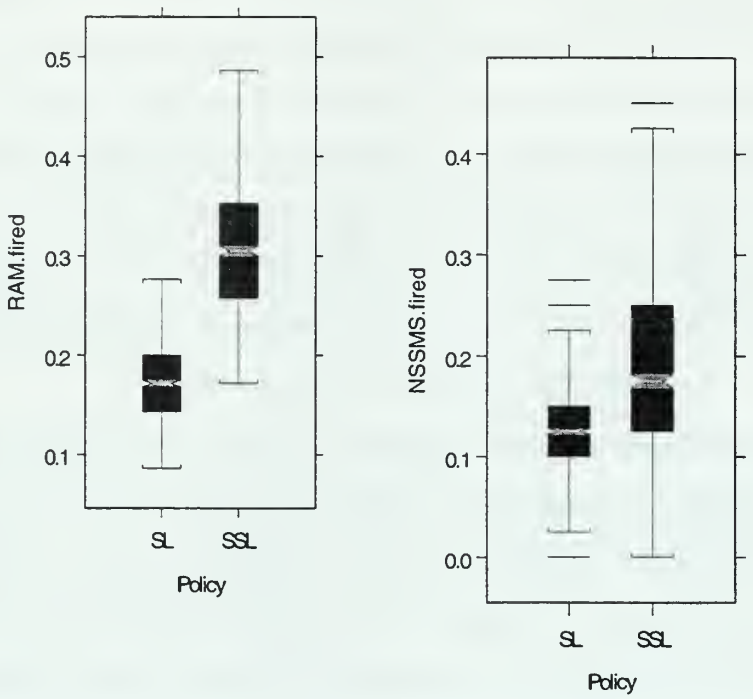

Figure 6.6. Boxplot of Fired SAMs over the Two Policies

After studying Figure 6.6 there should not be any doubt that the consumption of surface-to-air missiles is highly dependent on the firing policy in force. The non-overlapping notches clearly indicate a statistically significant difference in the means.

The next step was to compute the ratio of SAMs fired to SSMs shot down by SAM and to compare the different ratios corresponding to the two policies. Again we used the ratio of successful SSM rather than the actual number hitting the target, and the average of the ratio of consumption of NSSMS and RAM with equal weight. The observations are presented in Table 6.7 and Figure 6.7.

Policy	SL	SSL
Mean ratio of SAM pr SSM shot down by SAM	0.4829	0.6413

Table 6.7. Ratio of SAM Consumption per SSM Shot Down

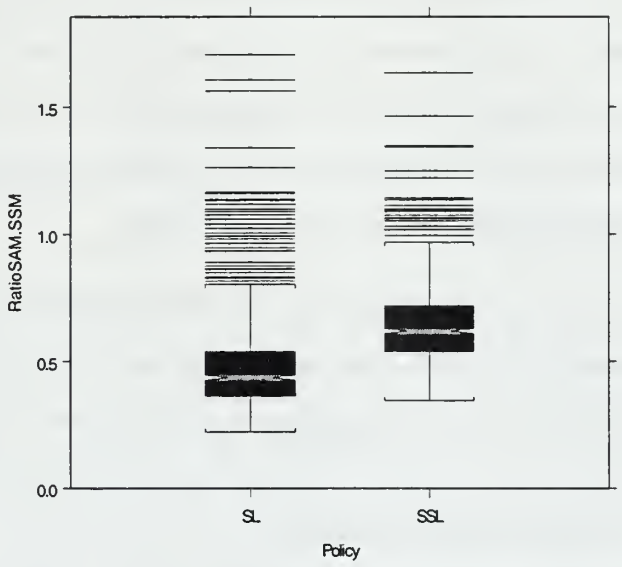

Figure 6.7. Boxplot of Success Ratio for SAM over Firing Policy

The data in Figure 6.7 are heavily skewed right, but it is nevertheless fair to say that the two policies have significantly different means. In this comparison a lower number is better; the defender would want to spend as little of his precious cargo of SAMs as possible yet still defend his ships. The OTC will have to make the tradeoff between an offensive and costly firing policy versus a restrictive policy that is clearly more efficient per weapon. A possible increase in the survivability of the defending ships in an upcoming attack must be compared to what the situation may be if the force is attacked again later and the availability of resupply.

We may state the following partial conclusions based on the discussion above. a) In the `kitchen` model there is a significant difference in the effectiveness of the firing policies SSL and SL, with the policy SL obtaining most kills per weapon. b)

The expected total number of SSMs shot down under the two different policies is significantly different under the assumptions of having good knowledge of the performance of the defender air defense. That is, "Policy" proved significant when "Performance" was fixed at one of its levels. In this sub-section we have been discussing choice of action taken by the defenders, that is at the "opposite" side from the previous section. Because the considerations over firing policy here is done by the executor of the air defense it must be fair to assume knowledge of performance of the defense systems. Hence we may conclude that from the defenders point of view, choice of firing policy is a significant contributor to the expected number of SSMs that hit their target.

2. SSM Effectiveness Versus Salvo Size

Among the first principles of attacking with SSMs, as discussed in Chapter 1, is to saturate the defender air defense. There are several ways to achieve this saturation, and normally it is achieved by use of a combination of techniques. One way to have saturation is to increase the number of weapons launched by firing a considerable higher number of missiles than is required to hit the target.

In this secondary experiment we used one of the two most efficient geometries from the main experiment, the one with minimum spread and minimum distance. The scenario was repeated 10 times, reducing the number of launched SSMs by three for each repetition. The variations in "Policy" and "Performance" were the same as under the main experiment, and again there were 45 replications for each sub plot. The results follows in Table 6.8, pictured in Figure 6.8.

SSM fired	30	27	24	21	18	15	12	9	6	3
Mean no hits	9.963	9.033	7.100	5.400	3.917	2.900	1.267	0.750	0.367	0.067
P(hit)	0.332	0.335	0.296	0.257	0.218	0.193	0.106	0.083	0.061	0.022

Table 6.8. Observations with Fixed Geometry, Varying Number of SSMs Launched

We may observe that the hit-probability of a given SSM in a salvo is dependent on the salvo size. The probability of having a successful missile drops faster than the number

of missiles in the salvo. There is apparently a synergy effect in launching a high number of SSMs in a salvo. Firing only 12 missiles, say, will saturate the air defense on the target ships to a moderate level while firing a large salvo of 30 SSMs increases the saturation and number of hits goes up more than proportionally to the salvo size. Doubling the salvo size from 15 to 30 increases the expected number of hits by a factor of 3.4.

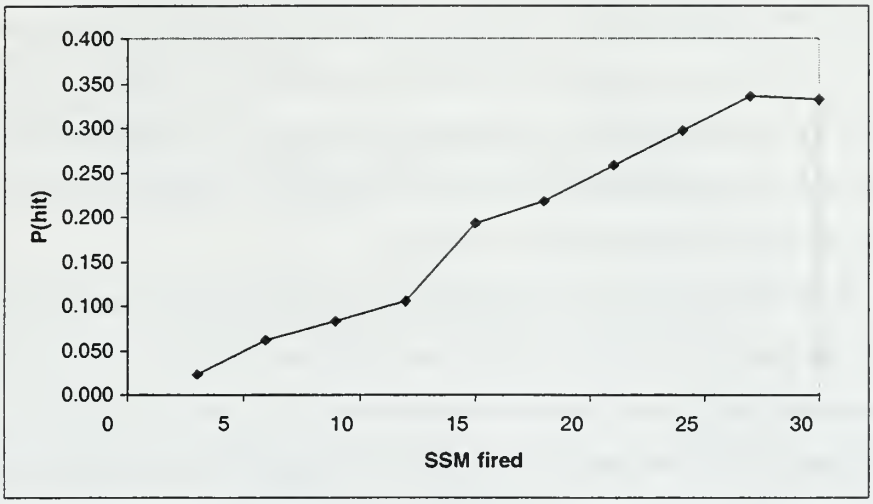

Figure 6.8. Relationship between Salvo Size and Hit-Probability

The number of SSMs that successfully hit their target in a given salvo is critically dependent on the size of the salvo and the level of saturation that is achieved in the defender's air defense. Launching a salvo of SSMs that is too small to create any level of saturation may wasting resources by the attacker.

3. Attacking with Higher Time Concentration

Along the same lines as the previous sub-section, time-concentration of the SSM attack is also a way to achieve saturation of the air defense onboard the defending ships. In the main experiment the delay between two consecutive firings from one missile launcher was set at 2.5 seconds. If all the missiles followed the same path to the target area and a shooter is about to fire 8 SSMs the launching process will take 20 seconds. This is the situation for two of the four attackers in the main experiment. There are two ways of reducing the time between the first and the last missile in an attack given that we

are launching all missiles along the same route: we may increase the number of launch platforms or we may reduce the time between consecutive firings.

Neither of these possibilities is generally available to the attacker; we would expect that she always would fire as rapidly as her systems allowed and that she also would employ all her possible launchers. For the analysis done in this thesis, however, it is important to investigate the leverage of this parameter of delay, as it is one estimated by the author. When running the secondary experiment to investigate the sensitivity to changing the interfiring delay, we reran the geometries with minimum spread and both long and short distance but with interfiring delay reduced from 2.5 seconds to 1.0 seconds. No other parameters were changed.

To investigate this issue a new set of data was collected consisting of the whole plots for long and short distance with minimum spread from the main experiment and from a new run with exactly the same parameters apart from the interfiring delay for the SSMs. While the main experiment was done using an interfiring delay of two and a half second; the new datapoints are collected with a delay of one second.

Again we will start the discussion by looking at boxplots, shown in Figure 6.9.

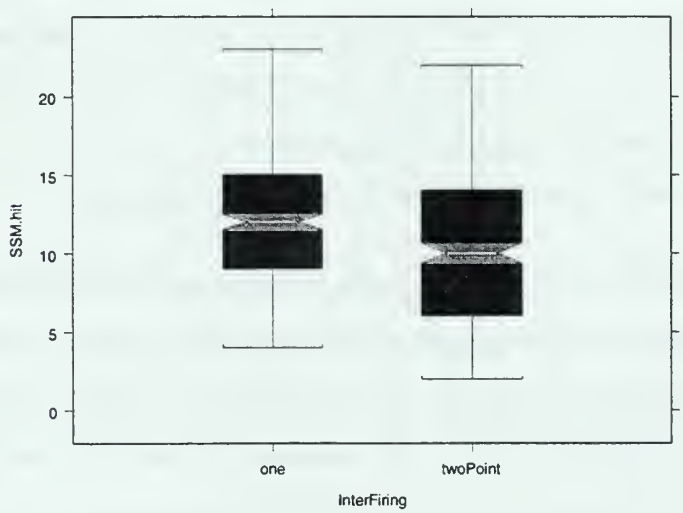

Figure 6.9. Boxplot for SSM Hits over Interfiring Delays 1 and 2.5 Seconds

From the picture it appears likely that the effect of compressing the time for launching the missiles is significant. The shorter delay appears to be associated with the most efficient launch procedure. We will again verify the graphical impression with an analysis of variance.

```
summary(aov(SSM.hit ~ InterFiring + Distance + Performance + Policy +
Error(Performance:Policy),data = rapid))

Error: Performance:Policy
            Df Sum of Sq  Mean Sq  F Value     Pr(F)
Performance  1  8848.022 8848.022 139.1077 0.0538477
Policy       1   259.200  259.200   4.0751 0.2928050
Residuals    1    63.606   63.606

Error: Within
             Df Sum of Sq  Mean Sq  F Value     Pr(F)
InterFiring   1   748.272 748.2722 132.3883 0.0000000
Distance      1     1.250   1.2500   0.2212 0.6383036
Residuals   714  4035.600   5.6521
```

Table 6.9. Function Call and ANOVA Table with Reduced Inter Firing Delay

From the ANOVA in Table 6.8 it is clear that the effect of reducing the inter-firing delay is beneficial to the SSM shooter. This is consistent with our expectations; we have implicitly assumed in the introductory chapters that a high rate of launching SSMs would contribute to the level of saturation of the air defense onboard the defending ships.

With a given scenario, a high rate of launching SSMs is more efficient than a low rate. The shortened time from first to last missile arriving in the target area contributes to the saturation of defender air defense.

4. Attacking at Very Short Distance

We have repeatedly argued that time is critical when the ships are defending themselves. Among other factors, the available time between detection and impact was assumed to have significance. After stating this, why does the model present only insignificant differences in number of successful SSMs over the three different launch

ranges? After all, the defenders have considerably more time available when the missiles are fired from 60 nautical miles than when launched at about 20.

In the sub-experiment to investigate this aspect, we reran the short-distance minimum-spread geometry but with an extra short distance. In the new setup, the launch distance was about 4.5 nautical miles, well inside the maximum range of the NSSMS (8.0 nautical miles) but also inside the range of the RAM (5.2 nautical miles) surface to air missiles. The new data were combined with the corresponding whole-plot from the main experiment, with engagement range approximately equal to 8 nautical miles. The boxplots are shown in Figure 6.10.

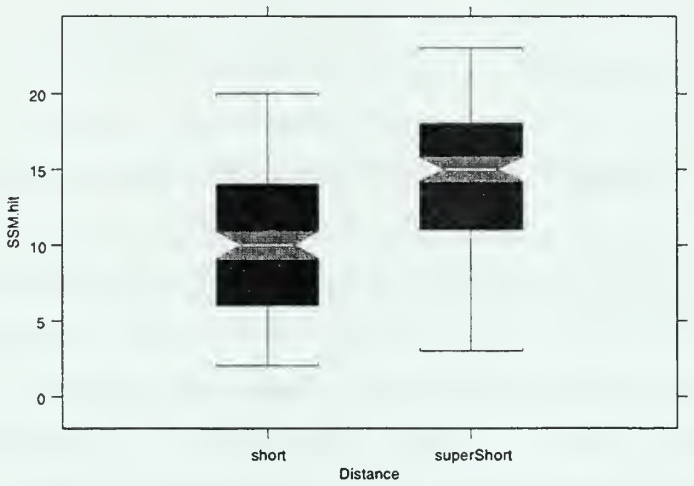

Figure 6.10. Boxplot for Hits over Short and Super-Short Distances

This is our first indication so far that the launch distance does affect the number of hitting SSMs. It is also the first geometry that has the launch positions well inside the range of the SAM. From the picture, it appears the difference is significant. Next we will verify this by an ANOVA, shown in Table 6.10.

```
summary(aov(SSM.hit ~ Distance + Performance + Policy +
                      Error(Policy:Performance), data = sShort))

Error: Policy:Performance

            Df Sum of Sq  Mean Sq  F Value     Pr(F)
Performance  1  5040.025 5040.025 123.9266 0.0570340
     Policy  1    73.803   73.803   1.8147 0.4065295
  Residuals  1    40.669   40.669


Error: Within

           Df Sum of Sq  Mean Sq  F Value Pr(F)
 Distance   1  1791.136 1791.136 304.0699     0
Residuals 355  2091.142    5.891
```

Table 6.10. Function Call and ANOVA for SSM Hits at Short and Super-Short Range

The ANOVA confirms the inferences drawn based on the boxplot. When comparing the number of SSM hitting the target for ranges 8 and 4.5 nautical miles respectively, there is a significant difference in the means.

Firing distance has proven not significant for all geometries with launch distances outside the maximum range of the surface-to-air missiles. With launch range shorter than SAM range, however, distance appears as a significant factor. And the reduction in range makes the number of successful SSMs go up.

5. Air-Defense with Long Range SAM

In the above sub-section we found the first indication that launch distance does have influence on the number of hits. We explored this further by rerunning the short-distance minimum-spread and long-distance minimum-spread whole plots exactly as in the main experiment, apart from the range of the NSSMS missile system.

This secondary experiment had the maximum range of the semi-active SAM increased from 8 nautical miles in the main experiment to 24 under the new scenario. Although the new parameters used for the SAM probably describe a weapon that does not exist in any arsenal, the scenario will prove a point of principal interest. If the results of

this experiment are in line with the above partial conclusion we would expect to see distance appearing as a significant factor again. This is because at long range the SSMs are launched from outside SAM range, while at short range they are launched inside the SAM coverage area. Again there are 45 observations in each sub-plot, totaling to 180 observations at each range.

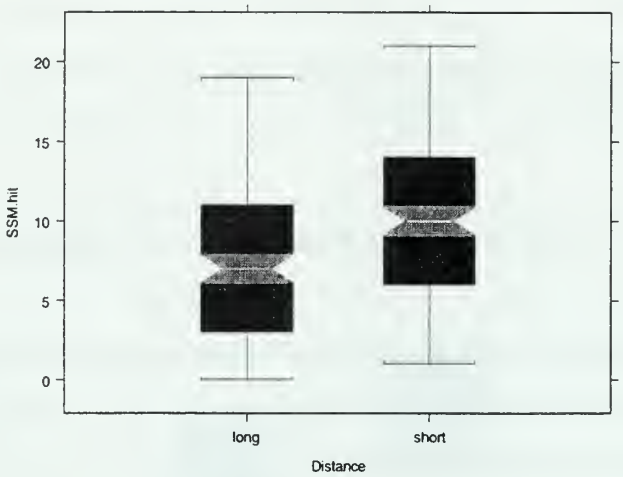

Figure 6.11. Boxplot for SSM Hit over Long and Short Distance with Extended SAM Range

```
summary(aov(SSM.hit ~ Distance + Performance + Policy +
            Error(Policy:Performance), data=Sam24))

Error: Policy:Performance
            Df Sum of Sq  Mean Sq  F Value     Pr(F)
Performance  1  5251.736 5251.736 165.1345 0.0494410
     Policy  1   186.336  186.336   5.8591 0.2494096
  Residuals  1    31.803   31.803


Error: Within
           Df Sum of Sq  Mean Sq  F Value Pr(F)
 Distance   1   686.136 686.1361 104.3615     0
Residuals 355  2333.986   6.5746
```

Table 6.11. Function Call and ANOVA for Secondary Experiment with Extended Range SAM.

Figure 6.11 and Table 6.11 show that distance is a significant factor in the secondary experiment with extended SAM range. With a SAM range 24 of nautical miles and SSMs launch ranges from about 60 and about 8 nautical miles, the shorter distance is advantageous to the SSM shooter. The same geometries but with a SAM range of 8 nautical miles showed no significance to SSM launch distance.

VII. CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

The main question asked in this thesis was: "Does the distribution of firing directions in a coordinated SSM attack have an effect on the expected result of the attack?" The question was addressed in the main experiment, and the answer was yes, there is a difference. Over all scenarios evaluated under the `kitchen` model, a minimum spread of the firing directions was advantageous to the SSM shooter. Apparently the best tactic for the officer in tactical command of a surface-to-surface missile attack is to locate her missile launchers, or use waypoints, to have all missiles arrive in the target area along the same bearing.

The starting point for this investigation was the differences in tactics used by the Norwegian Air Force and the Norwegian Navy when employing the Penguin SSM. Under the model designed and used in this thesis, the tactics used by the Air Force is the more efficient of the two.

We have also used the `kitchen` model to investigate some secondary questions, and the answers of interest to the attacker are:

- Decreasing firing distance is beneficial only if it can be reduced within SAM range. If all optional distances are outside SAM range, a reduction in launch distance proved to have no significance.

- Rate of SSM launch is important and reduction in the time between missiles arriving in the target area will increase the efficiency of the attack.

- The expected number of hits goes up more than proportionally to the salvo size. The marginal payoff for each extra missile in a salvo is at is largest when air defense starts to get saturated. The synergy of many SSMs in a short time increases the level of saturation of defender air defense.

The following observations are available in the `kitchen` model and should be of interest to the defenders:

- Choice of SAM firing policy does affect the quality of the air defense. Policy SL has the best success ratio per missile while policy SSL gives the best protection of the ships. Policy SSL implies a considerably higher spending of

weapons and factors not modeled in this thesis should be considered before making recommendations on policy.

- The maximum range of the SAM systems are, as anticipated, of critical importance to the air defense.

The `kitchen` model used several simplifying assumptions that could be generalized. In the process of designing the model as well as running the experiments, ideas for further improvements and research have materialized. Some of the improvements to the software package would be easily implemented while others would require both thorough studies of the existing code and careful redesign.

The following are suggested for improvement of the `kitchen` package, listed in anticipated sequence of increasing requirements of the programmer.

- Vectors to describe positions and velocities should be of the same dimension. As of now `modkit.util.spatial.Coor3D` is used for velocities while `...Coor4D` is used for positions. The recommendation is to use `Coor4D` for both.

- The detection factor, α, is given as a parameter to the `Radar`. This works only under the assumption that all possible targets have the same radar cross-section. In our case we had only one type of target and nothing was lost, but the detection factor should not be a feature of the radar but of the `RadarMediator`. This would facilitate multiple types of targets with different factor of detection.

- The CIC will order a new course to open arcs as necessary to engage with weapons. It will however not adjust the speed of the ship to reach the highest rate of turn possible. In the experiments in the thesis the ships have been set with an optimal speed, but the `CIC` could have done this.

- The selection of the best suited subordinate ship to handle a threat is using fixed relative values for range to target, number of SAM systems etc. The user should be given the opportunity to adjust the relative weight of the different factors.

- If we would want to model a ship with two significantly different SAM systems, the package may not provide high enough resolution. If this is the case, a scheme parallel to what was described for selecting firing ship (see

availability paragraph) should be implemented for selection of the SAM battery on each ship.

- After the first hit, the air defense onboard the hit unit should be degraded. This could be done according to a random variate specified by the user; i.e., the first hit reduces range of radar and rate of fire from a gun by specified amounts.

- The sensors assume that the detection factor specific to a target is the same regardless of aspect angle. This is true only for perfect spheres, but is not a far-fetched assumption when the variations in aspect angles are small as in our experiments. To model other scenarios with, for example, ship formations dispersed over a much larger area, an aspect-angle dependent detection factor would become important.

- The target identification and the information distribution models assume high standards of the data-link systems and the operators. A degradation of the systems could be designed, allowing for possible double tracking of some contacts (two track numbers on the same target) and open for the possibility of a target being "forgotten" for some time before it is given a target number. This implementation would require very careful programming and substantial testing. Also, parameterizing such "sloppiness" would be challenging.

Given the present model or a model improved as suggested above, further experiments to investigate the following questions are recommended.

- Compare firing policy SL and SSL under conditions where policy SSL sometimes will deplete available resources.

- Under fixed attack geometries, compare variations of the ships formations.

- Investigate the influence of varying the time-distance limit for where self-defense is allowed and coordinated SAM use is abandoned.

- Run the same experiments with the model calibrated with the best parameters available, classified if necessary. This would drastically reduce the variance caused by the "Performance" factor.

APPENDIX. THE CLASSES IN THE KITCHEN PACKAGE

The following is a list of all the classes in the kitchen package as they were at the time of the experiment in this thesis. They are listed in alphabetical order and there is a reference to the text of the thesis for further descriptions.

`Bearing`
Ref: Sub-section IV-A-1
Utility class, giving bearing between two positions, the compass course corresponding to a specified velocity and a course to turn to for a ship in order to open arcs for weapon systems.

`CIC`
Ref :Section IV - D and E
Core class in the simulation package, receives sensor information, reports new and lost targets, executes engagements with guns and or SAMs, on orders from `OTC` or on own initiative as an act of self defense. Orders new heading for its ship to make weapon bear.

`ComplexOperator`
Ref: Not discussed
Utility class for the `SolveCubic` class, allows for some algebra with complex numbers.

`Contact`
Ref: Sub-section IV - D - 1
Placeholder for all new targets and the information associated with them. Has methods to solve geometric questions associated with objects in motion. Used as framework for the data-link procedures.

`ContactManager`
Ref: Sub-section IV - D - 1
Bookkeeper for the `CIC` instances, sort all the targets held by a specific unit. May return the target associated with a track-number and vice-versa. Updates the target list based on information from the `CIC`.

`DataLinkEvent`
Ref: Sub-section IV - D - 3
Event dispatching class. Used to pass target information between `CIC`s and the `OTC` and vice versa.

`FPB`
Ref: Sub-section IV - E - 1
Missile launcher. Takes a time for when it's missiles should have impact, launches according to own and target position along specified routes.

Gun
Ref: Sub-section IV - E - 5
Generic gunnery system. Has a rate of fire, a specified number of rounds available and a rate of refill. Needs support from a `Tracker` to be able to open fire.

GunEvent
Ref: Sub-section IV - E - 5
Event dispatching class. Guns inform the listeners about the outcome of a gun-engagement, hit or not.

GunMissileObserver
Ref: Sub-section IV - E - 5
Neutral instance. Uses summation of distance-dependent probability of continuos misses in a gun-engagement to calculate time for hit if any.

MediatorFactory
Ref: Sub-section IV - E - 1
Utility class. Connects a SSM with the sensors by constructing instances of the `MoverMediator` class.

MissileEvent
Ref: Section IV - E
Event dispatching class. Missiles inform the listeners about the outcome of SAM and SSM engagements. Uses an explaining keyword (String).

ObserverEvent
Ref: Sub-section IV - E - 5
Event dispatching class. The `OutcomeObserver` informs the SAM and its target about the outcome of their encounter.

OTC
Ref: Section IV - D and E
Superior class on the defending side. Coordinates the use of force weapons when time allow, in charge of the data-link procedures .

OutcomeObserver
Ref: Section IV - E
Neutral instance. Judges the outcome of SAM versus SSM encounters.

Parameter
Ref: Not discussed
Utility class. Produces the random variables used by all other classes.

Radar
Ref: Section IV - C
Generic radar system. Holds radar-system data available for the neutral detection-instances. Maintain a list of tracked targets.

RadarMediator
Ref: Section IV - C
Neutral instance. Calculates time for when a target is detected or lost by a sensor. Uses radar-system parameters and geometry methods in numeric integration according to a non-homogenous Poisson process.

SAMBattery
Ref: Sub-section IV - E -4
Launcher of semi-active SAMs. Launches a salvo of `SAMissiles` when ordered and after starting to achieve continuos target information through a `Tracker`.

SAMBatteryPassive
Ref: Sub-section IV - E -4
Launcher of passive SAMs. Launches a salvo of `SAMissilePassive` when ordered.

SAMissile
Ref: Sub-section IV - E -4
Generic Surface-to-Air Missile. Will tailchase (or intercept) flying targets. Needs continuos knowledge of its targets position and velocity received through a `Tracker` if semi-active through internal sensors if passive.

Sensor
Ref: Section IV - C
Generic pure cookiecutter sensor.

SensorEvent
`Ref: Section IV - C`
Event dispatching class. `Sensor` informs listeners about new targets or existing targets lost.

SensorMoverMediator
Ref: Section IV - C
Neutral instance. Calculates time for when a target is detected or lost by a sensor. Uses pure cookie-cutter procedures.

SOFPB
Ref: Sub-section IV - E - 1
Coordinator of the `SSMissile` launchers. Takes a time for when the missiles should have impact, passes this information along to the subordinate `FPB` instances. Stops

and restarts the simulation when all the ordered SSMs in an engagement are shot down, have hit their target or are being lost in any other way.

SolveCubic
Ref: Not discussed
Utility class. Solves up to third order polynomials analytically. Used extensively by the methods solving geometric problems.

SpaceMover
Ref: Sub-section IV - B - 2
Generic 3D - mover. Extends SurfaceMover, has no means of modeling 2D movements by own methods.

SSMissile
Ref: Sub-section IV - E - 1
Generic Surface-to-Surface Missile. Will intercept a target on the surface, with or without traversing a series of waypoints enroute. Is equipped with an internal sensor that normally detects the target. The SSM is blind when the target is not detected while it is updated on, and compensates for, any change of velocity by the target after detection.

SSMRoute
Ref: Not discussed
Utility class. Holds a list of waypoints and a specific reference to the target for `SSMissiles`.

SteadyVelocityEvent
Ref: Section IV - C and D
Event dispatching class. Any mover will inform listeners about new velocity or new altitude by this method. Uses a keyword (String) to distinguish between different statevariables.

SurfaceMover
Ref: Sub-section IV - B - 1
Generic 2D - mover. Core class for all moving objects. Holds methods for a variety of different procedures for motion in two dimensions. Changes of heading or speed are done according to defined rates or instantly as decided by the user. Holds two advanced methods for interception.

Tracker
Ref: Sub-section IV - E - 3
Generic target illuminator. Takes one target and will after delay report that the target is tracked. Necessary for engagements with guns and semi-active SAMs. Requires line of sight to the target.

`TrackerEvent`
Ref: Sub-section IV - E - 3
Event dispatching class. `Tracker` informs listeners about gained or lost track of a target.

`TrialShort`
Ref: Chapter V
Pure execution class. Holds three geometries at short distance and all the parameters as described in Section V - 2.

`TrialIntermediate`
Ref: Chapter V
Pure execution class. Holds three geometries at intermediate distance and all the parameters as described in Section V - 2.

`TrialLong`
Ref: Chapter V
Pure execution class. Holds three geometries at long distance and all the parameters as described in Section V - 2.

LIST OF REFERENCES

Arntzen, A., *Software Components for Air Defense Planning,* Master's Thesis in Operations Research, Naval Postgraduate School, Monterey, California, 1998

Bowditch, N., *American Practical Navigator,* Defense Mapping Agency Hydrographic Center, 1977

Chan, P., Lee, R., and Kramer, D., *The Java Class Libraries*, Addison-Wesley 1998

Gosling, J., Joy, B., and Steele, G., *The Java Language Specification,* Addison-Wesley, 1996

Hinkelmann, K., and Kempthorne, O., *Design and Analysis of Experiments*, Volume 1, John Wiley & Sons, Inc 1994

Jane's *Fighting Ships 1998 - 1999,* Jane's Information Group 1998

Jane's *Naval Weapon Systems 1970 - 71*, Janes's Information Group 1970

Jane's *Naval Weapon Systems* 1989 -, Jane's Information Group 1989 -

Law, A. M.and Kelton, W.D., *Simulation Modeling & Analysis,* 2nd ed., McGraw-Hill, 1991

Washburn, A. R., *Search and Detection*", 3rd ed, Chapter II, INFORMS, 1996

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J Kingman Rd., STE0944
   Ft Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Professor Arnie H. Buss, Code OR/Bu 1
   Department of Operations Research
   Naval Postgraduate School

4. Professor James N. Eagle, Code OR/Er 1
   Department of Operations Research
   Naval Postgraduate School

5. Professor Robert R Read, Code OR/Re 1
   Department of Operations Research
   Naval Postgraduate School

6. Education Branch , Defense Command Norway 1
   Oslo mil - Huseby
   N 0016 OSLO, Norway

7. Above Water Warfare School 1
   Norwegian Naval Training Establishment
   N 5078 HAAKONSVERN, Norway

8. Norwegian Naval Academy 1
   P.O. Box 54
   N 5164 LAKSEVAAG, Norway

9. Norwegian Defense Research Establishment 2
   System Analysis Division (1), Electronics Division (1)
   P.O. Box 25
   N 2007 KJELLER, Norway

10. Inge Andre Utaaker 3
    Vaakleivbrotet 1
    N 5155 BONES, Norway